

Johannes Kepler Universität
Institut für Wirtschaftsinformatik
Communications Engineering
Freistädter Straße 315
A-4040 Linz / AUSTRIA
Phone: +43/732/2468-7102

Zerlegung von digitalisierten Lehrbüchern für den Aufbau eines Contentpools

Diplomarbeit

Zur Erlangung des Titels „Magister der Sozial- und Wirtschaftswissenschaften“
(Mag. rer. soc. oec.)

In der Studienrichtung Wirtschaftsinformatik

Verantwortliche Betreuer:

o. Univ.-Prof. DI Dr. Christian Stary

Dr. Andreas Auinger

Autor:

Stefan Berger

Linz, 26. Mai 2004

Eidesstattliche Erklärung

Ich erkläre an Eides Statt, dass ich die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und alle den benutzten Quellen wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Linz, 26. Mai 2004

Stefan Berger, 9956698

Zusammenfassung

Während Wissenstransfer-Anwendungen sich immer größerer Beliebtheit und Verbreitung erfreuen, vermissen deren Benutzer bis jetzt noch die Möglichkeit zur selbständigen Gestaltung dynamischer Lernunterlagen. In der vorliegenden Arbeit wird, basierend auf dem aktuellen Stand der Technik, ein Vorschlag erarbeitet, auf welche Weise die bis dato statischen Lernunterlagen durch dynamische Erweiterungen aufgewertet werden können.

Als Grundlage für die Arbeit dienen XML Topic Maps. Sie dienen zur Erstellung semantischer Verknüpfungen als „Rückgrat" eines Pools aus zerlegten Lehrbüchern, dem so genannten „Contentpool". Der Autor zeigt an Hand eines Projekts, das nach dem Software Life Cycle Modell in die Phasen Analyse, Entwurf, Implementierung und Testen gegliedert war, den Weg zur Entwicklung von Contentpool-Werkzeugen auf. Wichtige Meilensteine und Ergebnisse der Projektphasen finden in der Diplomarbeit Berücksichtigung.

Mit Hilfe der Contentpool-Werkzeuge entsteht aus elektronischen Büchern, Zeitschriften, wissenschaftlichen Abhandlungen usw. durch Zerlegung und Aufbereitung mit Metadaten ein Speicher aus einzelnen „Wissens-Atomen". Diese bilden das Fundament, auf dessen Grundlage sich Lehrende und Lernende nach persönlichen Präferenzen Unterlagen zur Unterstützung des virtuellen Lehrens und Lernens zusammen stellen können.

Abstract

Telelearning environments (i.e. software supporting learning in virtual class rooms) are getting more and more important nowadays. Users of those systems may expect electronic material supporting the learning process to be provided by the teacher, for instance. However, all material provided for learning purposes is completely static so far. What, in most cases, is missing within telelearning environments is the possibility, be it for the teacher or the student, to create learning materials dynamically. This diploma thesis contains a proposal for how to satisfy these demands based on telelearning state-of-the-art.

Based upon the technical concept of XML Topic Maps, the author will present a way how to create a semantics-based repository of so called "knowledge atoms" that can be created by decomposition of and metadata creation on slices of electronic documents (books, papers, scientific magazines, and so on). A knowledge atom repository, as mentioned above, will provide teachers and learners with the possibility of creating individual learning materials, tailored for their personal needs.

Conclusions won within this diploma thesis have yet been implemented in several tools. The diploma thesis itself is available in German language only.

Hinweis zur geschlechtsneutralen Schreibweise

In der vorliegenden Diplomarbeit habe ich darauf verzichtet, Formulierungen zu verwenden, die explizit beide Geschlechter ansprechen. Eine solche Schreibweise erachte ich persönlich als umständlich, da die dabei verwendeten Sprachkonstruktionen den Text unnötig aufblähen. Aus diesem Grund habe ich versucht, meinen Text möglichst geschlechtsneutral zu formulieren. Überall, wo mir das nicht gelungen ist, habe ich, wie bisher üblich, die männliche Form verwendet. Damit möchte ich jedoch stets, sofern nicht explizit auf einen anderen Sachverhalt hingewiesen wird, beide Geschlechter ansprechen.

Stefan Berger

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

„Unsere Kulturgeschichte ist eine Geschichte des Wissens. Sie ist eine Geschichte der Funktion und Handhabung von Wissen […] und letztlich des Triumphzuges der Wissenschaft.“ ([SchmZuck, 2003], S. 30)

„Die ‚Knowledge Economy' ist mittlerweile von der Trendvokabel zur Realität befördert worden. Bereits 50 % des Bruttosozialprodukts werden laut einer aktuellen OECD-Studie[1] *von ‚knowledge based' Unternehmen erwirtschaftet.“* ([SchmZuck, 2003], S. 7)

Die oben zitierten Aussagen zeigen: die Gesellschaft in den führenden Industrienationen der Erde, so auch jene in Österreich, hat sich in den vergangenen Jahren und Jahrzehnten von einer Industrie- zu einer Informationsgesellschaft gewandelt. Der Stellenwert von „Wissen“ als Produktionsfaktor hat radikal an Bedeutung dazu gewonnen. Diese Entwicklung hat bis jetzt noch kein Ende gefunden. Zurzeit ist der nächste Wandel im Gang, denn mittlerweile befinden wir uns im **Übergang von der Informations- zu einer Wissensgesellschaft**. Mit anderen Worten: „Wissen“ wird in den fortschrittlichsten Volkswirtschaften der Erde (z.B. Staaten der Europäischen Union, USA, Japan) in der Zukunft zum wichtigsten Produktionsfaktor für die Gesellschaft werden [SchmZuck, 2003].

Ein höherer Stellenwert des Wissens führt zu einem höheren Bedarf zur **Vermittlung** von Wissen. Die **Bildungssysteme** in den Industrienationen werden gefordert sein, mit der erhöhten Nachfrage an Wissensvermittlung Schritt zu halten. Mit den Mitteln des klassischen Unterrichts wird der zukünftige Bildungsbedarf nur schwer zu decken sein (vgl. [Dodds, 2001a], S. „1-17“ ff.). **Neue Konzepte für den Unterricht** in Schulen und Universitäten sind daher gefragt.

Virtuelle, computerbasierte Lernumgebungen können dafür Abhilfe schaffen (vgl. [Fletcher, 2001], zitiert in [Dodds, 2001a], S. „1-18“ f.). Bei virtuellen, computerbasierten Lernumgebungen oder **Wissenstransfer-Umgebungen** handelt es sich um Computerprogramme, die einen **Lehrer unterstützen** oder sogar ersetzen können. Mit deren Hilfe kann an die Stelle des Lehrers ein Computer treten, wobei die Abwicklung des Unterrichts nicht einmal mehr in einem traditionellen Klassenzimmer erfolgen muss. Unter Miteinbeziehung des Internets können so genannte „**virtuelle Klassenräume**“ entstehen. Zu den obigen Ausführungen wird zusätzlich auf das Glossar (Abschnitt 10.2, Seite 333 ff.) verwiesen.

Die vorliegende Arbeit liefert einen Vorschlag zur Erweiterung der Funktionalität von Wissenstransfer-Umgebungen, wodurch diese zusätzlich zur Fähigkeit der Vermittelung von Wissen die Fähigkeit zur **Speicherung von Wissen** erhalten. Es wird der im Gegensatz zu bisher realisierten Wissenstransfer-Umgebungen wird neue Ansatz verfolgt, elektronische Dokumente bis auf die Ebene der **kleinsten, sinnvoll zusammen hängenden Einheiten von Wissen**, den so genannten „Wissensatomen“ (für eine exakte Definition

[1] Genauere Daten bezüglich der genannten OECD-Studie werden in der Quelle leider nicht angegeben.

vgl. das Glossar im Abschnitt 10.2, Seite 342), zu zerlegen und anschließend zu speichern.

Die Speicherung der Wissensatome erfolgt mit dem Ziel, bestimmte Teilmengen derselben letztlich zu **neuen elektronischen Unterlagen** zusammen zu bauen. Dabei fließen semantische Zusammenhänge zwischen den Wissensatomen in die Auswahl der Teilmenge entscheidend mit ein. Der neue Ansatz bietet den Benutzern dabei genug Spielraum, die aus den gespeicherten Wissensatomen erzeugten Lernmaterialien auf **persönliche Bedürfnisse** anzupassen.

Als Grundlage für das oben genannte Konzept der aus Wissensatomen zusammen gesetzten Lernunterlagen dient eine Art „**Wissensbasis**", die aus zerlegten elektronischen Dokumenten besteht. Die zerlegten Dokumente werden mit **Metadaten angereichert**, deren **semantische Zusammenhänge** erfasst und die dadurch entstandenen fertigen Wissensatome gespeichert. Den Lehrenden und Lernenden wird durch die Erstellung von Kursmaterialien und Nachschlagewerken aus Wissensatomen der **Zugriff** auf das gespeicherte Wissen ermöglicht.

Im vorliegenden einführenden Kapitel wird dem Leser der Inhalt dieser Diplomarbeit vorgestellt. Das Ziel der Diplomarbeit wird in der Folge exakt definiert (Abschnitt 1.1). Im Abschnitt 1.2 wird der Aufbau und Ablauf der Untersuchung zur Erreichung der vorgestellten Ziele behandelt. Das Einführungs-Kapitel wird mit Hinweisen zur Gestaltung und zum Aufbau des vorliegenden Dokuments (Abschnitt 1.3) abgeschlossen.

1.1 Ziele dieser Diplomarbeit

Das oberste Ziel dieser Diplomarbeit besteht darin, die dem Stand der Technik entsprechenden Architekturen virtueller, computerbasierter Lernumgebungen (in der Folge werden diese als „Wissenstransfer-Umgebungen" oder auch „Wissenstransfer-Anwendungen" bezeichnet; vgl. dazu auch das Glossar, Abschnitt 10.2, Seite 343) zu erweitern. Es wird für diesen Zweck ein Konzept für die effiziente und die Semantik berücksichtigende **Speicherung von Lern-Inhalten** entwickelt. Das genannte Konzept wird in dieser Diplomarbeit als „Contentpool" bezeichnet.

Wie aus der Bezeichnung „**Contentpool**" schon hervorgeht, ist dessen Zweck darin zu sehen, einen Pool von Lern-Inhalten zu speichern. Über Werkzeuge sollen den Benutzern von Wissenstransfer-Umgebungen die Lern-Inhalte zur Verfügung stehen. Die Lern-Inhalte des Contentpools werden in Datenstrukturen, den „**Wissensatomen"**, gespeichert. „Wissensatome" sind die kleinsten sinnvoll zusammen hängenden Einheiten von Wissen, die sich aus dem Inhalt von Lehrbüchern und sonstigen Unterlagen durch Zerlegung bilden lassen. Die Wissensatome können zu einem späteren Zeitpunkt nach Bedingungen, die über die Semantik der Inhalte definiert werden („semantische Bedingungen"), wieder zu Dokumenten zusammengesetzt werden. Dabei soll der ursprüngliche semantische Zusammenhang des Wissensatoms keine Rolle spielen.

Die **physische Speicherung** der zerlegten elektronischen Lehrbücher erfolgt in einer **Datenbasis**. Semantische Beziehungen zwischen den „Trümmern" der Dokumente werden in einem **semantischen Netz** gespeichert. Inhalte und semantische Beziehungen

sind folglich separat und unabhängig voneinander gespeichert. Datenbasis und semantisches Netz **zusammen** ergeben den Contentpool.

Die Grundidee des Contentpools ist vergleichbar mit der einer Wissensbasis. Todo: [Abgrenzung zu Wissensbasis, Referenzen auf Literatur zu wichtigsten Schlagwörtern.]

Für die Verwirklichung des Contentpool-Konzepts ist es notwendig, Möglichkeiten zur Zerlegung von Dokumenten einerseits und zur Speicherung zerlegter Dokumente andererseits zu finden oder zu entwickeln.

Wesentliche Zielsetzung für den Contentpool ist, den Benutzern von Wissenstransfer-Umgebungen die Möglichkeit zu eröffnen, sich aus den gespeicherten Inhalten (den Wissensatomen) dynamisch elektronische Lernunterlagen zu erstellen und zu gestalten. Dafür ist es notwendig, Werkzeuge zu designen, die effektiv und effizient den Zugriff auf Wissensatome im Contentpool ermöglichen. Das Konzept soll zudem Spielraum bieten, um die Lernunterlagen auf die speziellen **persönlichen Vorlieben** und **kognitiven Bedürfnisse** der Benutzer abzustimmen. Gegebenenfalls sollen die Lernunterlagen sogar mit eigenen Dokumenten des Benutzers ergänzt werden können.

Bis dato sind Wissenstransfer-Umgebungen auf **rein statische Materialien** zur Wissensvermittlung (in dieser Arbeit als „Lehr- und Lernmaterialien" bezeichnet) angewiesen (vgl. [Collier, 2002], [Comp, 2002], [IMS Content, 2001], [Sander, 2001], [Trial, 2003]). Die vorliegende Diplomarbeit stellt mit dem Contentpool-Konzept auch Architekturen für Werkzeuge zur Verfügung. Somit wird ein vollständiges Modell für die Erweiterung von Wissenstransfer-Umgebungen vorgestellt, das die Realisierung eines Contentpools aus zerlegten wissenschaftlichen Unterlagen zur **benutzerdefinierten** und **dynamischen** Generierung von Lehr- und Lernmaterialien ermöglicht.

Das vorgestellte Contentpool-Konzept sowie die Architekturen für unterstützende Werkzeuge wurden in einer Implementierung umgesetzt. Auch die Implementierung wird in der Diplomarbeit behandelt. Damit wurde die erforderliche technische Unterstützung zum Verwirklichen der genannten Ziele beispielhaft realisiert und im praktischen Einsatz erprobt.

Aus dem übergeordneten Ziel (Entwicklung eines Konzepts für einen Contentpool) lassen sich durch Zerlegung folgende Unterziele bilden:

- Die **Effizienz** des Wissenstransfers (Telelearning) und der Plattformen zur Unterstützung des Wissenstransfers ist zu verbessern.
- Die **Wiederverwendbarkeit** von Wissen, das in Lehrbüchern gespeichert und aufbewahrt wird, ist zu erhöhen.
- Lernende (Studenten) sind beim **Prozess des Lernens** zu unterstützen, indem ihnen individualisierte Kursmaterialen und personalisierte Nachschlagewerke zur Verfügung gestellt werden.
- Lehrende (Professoren und Lehrbeauftragte) sind beim **Prozess des Lehrens** zu unterstützen, indem ihnen das Editieren von Kursmaterialien und das Erstellen von Nachschlagewerken erleichtert werden.

- Die Speicherung von Wissensatomen im Contentpool hat den **Stand der Technik** bei Datenbasen und semantischen Netzen zu berücksichtigen. Das **Datenmodell** des Contentpools ist so zu entwerfen, dass der Speicherbedarf und die Zugriffszeit minimal sind, die Interoperabilität und Offenheit dagegen maximal sind.
- Um die Brauchbarkeit des Contentpool-Konzepts zu erhöhen, ist ein **Werkzeug** für die **Erzeugung von Wissensatomen** zu designen und zu implementieren.
- Die für den **Zugriff auf die Wissensatome** im Contentpool notwendigen **Werkzeuge** sind zu designen, zu implementieren und zu testen.

Die genannten Ziele weisen den Weg für die weitere Arbeit. Sie dienen unmittelbar zur Definition von zu lösenden Fragestellungen. Diese werden im Kapitel 2 (Ausgangssituation und Problemstellung, Seite 15 ff.) behandelt.

1.2 Ablauf und Aufbau der Untersuchung

Die in diesem Dokument dargelegten Ergebnisse und Erkenntnisse wurden im Rahmen des **Projekts „Contentpool-Entwicklung für Scholion WB+"** gewonnen. Gegenstand des (Gesamt-) Projekts „Scholion WB+" war die Entwicklung einer web-basierten Wissenstransfer-Umgebung mit der Bezeichnung „Scholion WB+". Das Projekt „Contentpool-Entwicklung" als Teilprojekt von „Scholion WB+" definierte das Ziel, einen so genannten Contentpool für die Wissenstransfer-Umgebung zu entwickeln.

Als Grundlage für das Gesamtprojekt dienten die Erkenntnisse aus der Entwicklung und Implementierung der Vorgängerversion der Scholion Teleteaching und Telelearning Software (vgl. [Ellmer et. al., 1998], [Froschauer et al., 1999], [Auinger, 1999], [Karrer, 2000]).

Damit ergibt sich für die Diplomarbeit ein starker praktischer Bezug. Mit den **theoretischen Grundlagen** beschäftigen sich die ersten Kapitel (2: Ausgangssituation und Problemstellung, 3: Auswahl der Konzepte und Technologien, 4: Related Work und 5: Bewertung der Konzepte und Technologien). Der Rest des Dokuments erläutert das **Design** und die **Implementierung der Software** sowie Erkenntnisse aus ersten **Tests** der Software[2] (Kapitel 6: Design der Software, 7: Implementierung der Software und 8: Test der Software). Am Ende des Dokuments befinden sich eine **Zusammenfassung** der wichtigsten Eckpunkte der Diplomarbeit (Kapitel 9: Fazit und Ausblick) sowie der Anhang (Kapitel 10).

Der Prozess zur Erstellung dieser Diplomarbeit war selbstverständlich stark an das Fortschreiten des Projekts „Contentpool-Entwicklung" gekoppelt. Bei der Entwicklung der Software wurde nach dem charakteristischen Vorgehensmodell des Software Engineering, einem evolutionären, iterativen **Phasenschema der Entwicklung** eines Systems (z.B. eines Software-Produkts) **vom Groben zum Feinen**, vorgegangen. Gemäß dem Vorgehensmodell werden die vier Phasen Analyse, Design, Realisierung und Einsatz un-

[2] Anmerkung: Mit dem Begriff „Software" sind sowohl der Contentpool als auch die Werkzeuge für den Zugriff auf die Wissensatome im Contentpool gemeint.

terschieden (vgl. [Oestereich, 1997], S. 63 – 73). Entsprechend dieser Unterscheidung wurde der Projektablauf von „Scholion WB+“ in folgende Phasen untergliedert:

- Analyse
- Design
- Implementierung
- Testen

Auf den kommenden Seiten folgen die Übersicht zum Ablauf der Untersuchung, deren Erkenntnisse in dieser Diplomarbeit dokumentiert sind (Abschnitt 1.2.1) sowie die detaillierte Beschreibung der Schritte, die während der Untersuchung ausgeführt wurden (Abschnitt 1.2.2).

1.2.1 Übersicht: Untersuchungsablauf

Zur Illustration wird das **Vorgehensmodell** zur Erlangung der in der Diplomarbeit behandelten Erkenntnisse vorgestellt und in Abbildung 2 gezeigt. Zuvor wird in Abbildung 1 die zur Modellierung verwendete **Symbolik** definiert und erläutert.

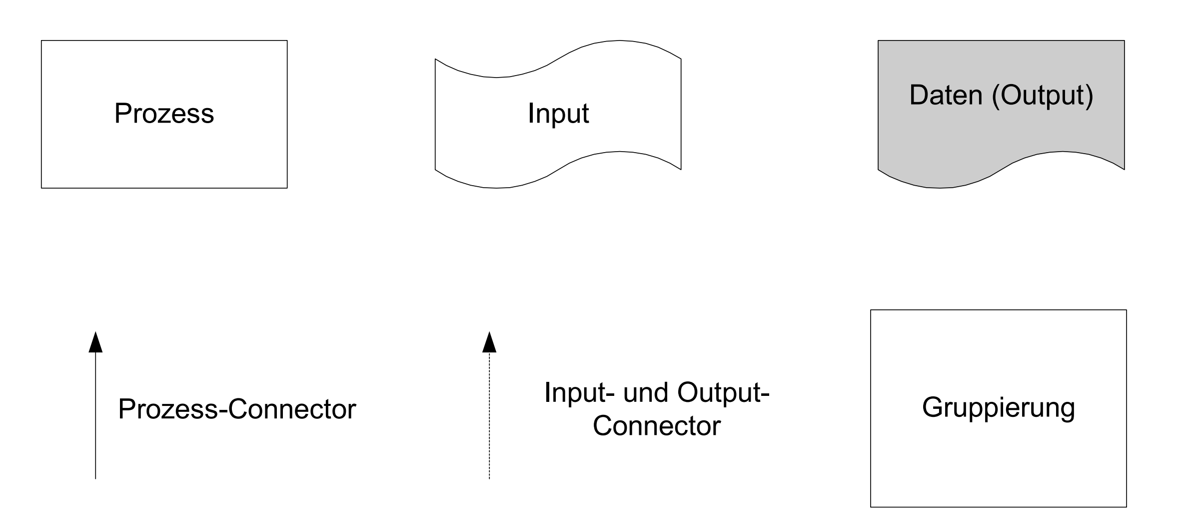


Abbildung 1: Notation zum Untersuchungsablauf

Die Elemente der Notation zum Untersuchungsablauf (Abbildung 1) werden wie folgt definiert:

- **Prozess**: Vorgang, der von einer oder mehreren Personen ausgeführt wird. Eine explizite Zuordnung von Rollen zu den Prozessen ist nicht notwendig, da im Rahmen der hier präsentierten Untersuchung (vgl. das Modell in Abbildung 2) stets der Autor als Rolle (ausführende Person) zugeordnet werden kann.
- **Input**: Daten, Quellen oder Objekte, die für die Untersuchung wesentliche unterstützende Inputs liefern, wie z.B. Theorien, Software, Methoden, Literatur etc.
- **Output**: dies sind Daten, die im Verlauf der Untersuchung erstellt wurden und eventuell auch weiterverarbeitet werden, wie z.B. Dokumente oder Software.

- **Prozess-Connector**: verbindet Prozesse untereinander. Die Richtung des Pfeils gibt die zeitliche Reihenfolge an, in der die Prozesse geschehen (müssen).
- **Input- und Output-Connector**: verbindet Prozesse mit Inputs oder Outputs.
- **Gruppierung**: designtechnisch verwendetes Element zur Gruppierung anderer Notationselemente. Die Gruppierung wird eingesetzt, um die Lesbarkeit eines Diagramms zu verbessern.

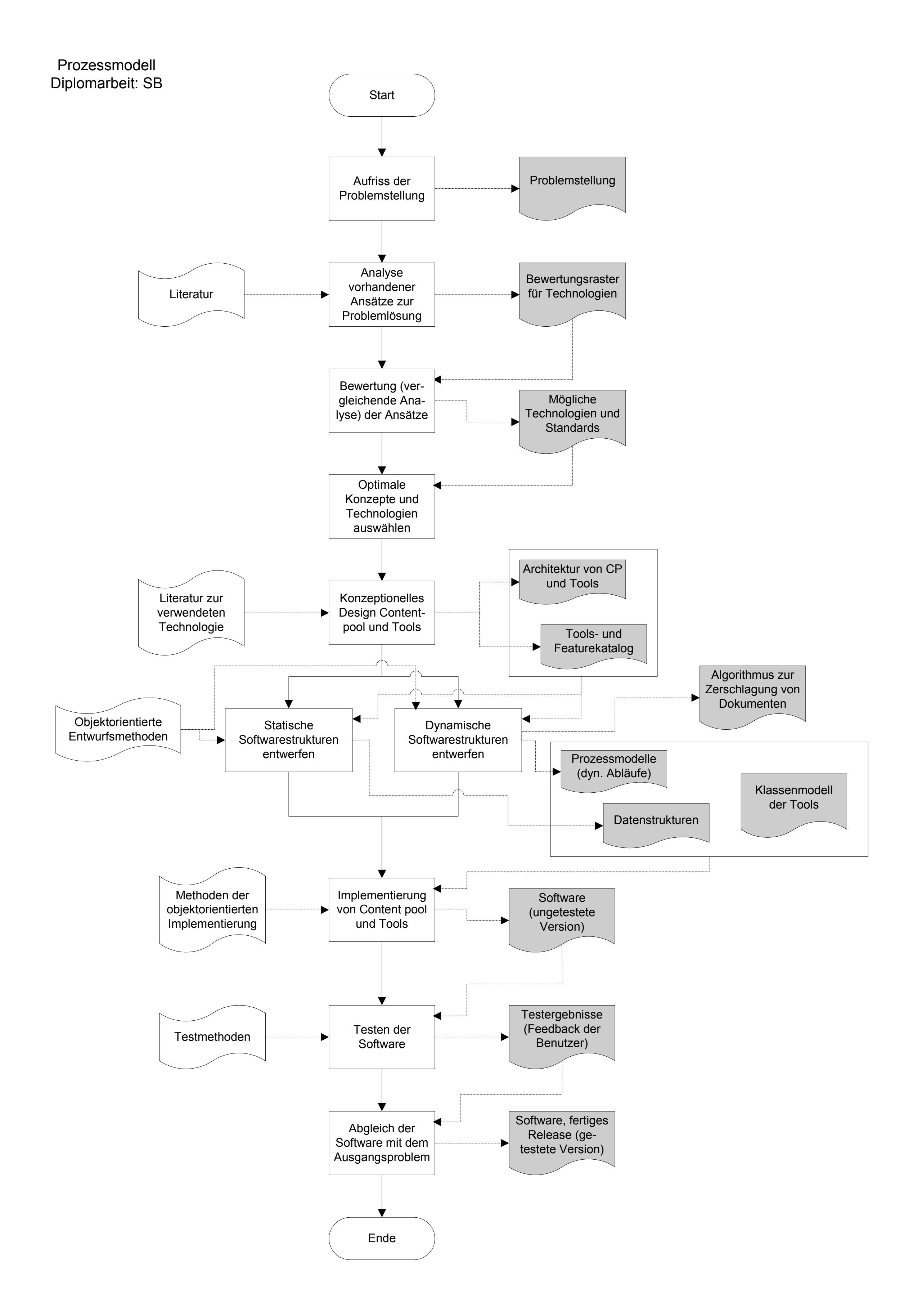


Abbildung 2: Untersuchungsablauf

1.2.2 Beschreibung des Untersuchungsablaufs

Nachfolgend werden die im Prozessmodell des Untersuchungsablaufs (siehe Abbildung 2) genannten Elemente (Prozesse, Daten, usw.) beschrieben.

1.2.2.1 Aufriss der Problemstellung

Zu Beginn ist der **Ausgangspunkt für die Fragestellungen** zu erarbeiten, welche im Verlauf der Untersuchung zu beantworten sind. Als Ausgangspunkt für die Entwicklung eines Contentpools können Ziele für die Unterstützung von Lehrenden und die Unterstützung von Lernenden identifiziert werden. (Vgl. dazu auch Abschnitt 1.1 Ziele dieser Diplomarbeit, Seite 2.)

Lehrende sind bei der Abhaltung von Kursen mit Hilfe von Scholion WB+, d.h. bei ihrer **Lehrtätigkeit in der Wissenstransfer-Umgebung**, zu unterstützen. **Lernende** dagegen sind beim Durcharbeiten der Materialien für die angebotenen Kurse und beim Lernen, d.h. bei ihrer **Lerntätigkeit in der Wissenstransfer-Umgebung**, zu unterstützen.

Allgemein, d.h. unabhängig von einer der beiden genannten Gruppen von Benutzern, sind die Merkmale selbstgesteuerter Wissenstransfer-Umgebungen zu beachten. Diese geben Rahmenbedingungen vor, die in die Problemstellung einfließen. A. Auinger und Ch. Stary nennen insgesamt vier Merkmale, von denen zwei für die Ziele der Diplomarbeit von Bedeutung sind (vgl. [AuingStary, 2003]):

- Unterstützung von **Individualisierung**: für die Konzeption des Contentpools ist wichtig, den Benutzern die Anpassung der Inhalte auf persönliche Bedürfnisse zu ermöglichen [AuingStary, 2003].
- Verknüpfung von **Inhalten und Kommunikation**: effektiver Wissenstransfer erfordert die Möglichkeit, Inhalte in Wissenstransfer-Umgebungen untereinander und mit Kommunikationselementen zu verknüpfen [AuingStary, 2003].

1.2.2.2 Analyse vorhandener Konzepte

Bei der Analyse vorhandener Konzepte steht das Ziel im Mittelpunkt, einen Überblick über den **Stand der Forschung und Technik** („state of the art") zur Speicherung von Dokumenten, zerlegten Dokumenten, Fakten über die Dokumente, das Wissen eines Autors von Dokumenten und so weiter zu erarbeiten. Kurz, der wissenschaftliche und praktische Hintergrund zum Thema Content Management wird abgesteckt.

Um einen Überblick zum State of the Art zu erarbeiten, werden zahlreiche Ansätze in Wissenschaft und Praxis untersucht. Zunächst interessiert nur die Frage, welche Konzepte für die Realisierung der gesetzten Ziele hilfreich sind. Anschließend werden aus auf dem Markt erhältlichen **Content-Management-Systemen** oder ähnlichen Produkten die zu Grunde liegenden Konzepte analysiert. Im Mittelpunkt des Interesses stehen dabei Ansätze und Projekte, die öffentlich und frei verfügbar sind. Dazu gehören z.B. international anerkannte offene Standards (wie die Auszeichnungssprache XML).

Die Auswahl der Konzepte (als Auszug aus kommerziell erhältlichen oder offen verfügbaren Produkten bzw. nur theoretischen Konzepten) wird dann mit rein theoretischen und

sonstigen Konzepten, Ansätzen, Spezifikationen, Architekturen, usw. vervollständigt, die noch nicht in einem Produkt umgesetzt wurden.

Abgerundet und ergänzt wird die Auswahl der bis dahin gefundenen Konzepte durch eine **Analyse der einschlägigen Fachliteratur**. Aus wissenschaftlichen Dokumenten werden Konzepte identifiziert, die sich mit Content Management oder ähnlichen Themen befassen.

Im Rahmen der Analyse vorhandener Konzepte wird (meist wissenschaftliche) **Literatur** als Input verwendet, um zu Erkenntnissen über gängige Ansätze, Technologien, Standards usw. zu gelangen. Als Output der Analyse wird ein Raster mit den Merkmalen für die Auswahl von Konzepten, Technologien, Standards usw. erstellt. Dies ermöglicht die methodisch korrekte Durchführung der Bewertung und anschließenden Auswahl von Konzepten.

1.2.2.3 Bewertung der Konzepte

Ist erst die Menge der in Frage kommenden Konzepte abgesteckt, sowie die Erstellung eines **Merkmalskatalogs** und der **Definition von Metriken** für die Bewertung (vergleichende Analyse) der Konzepte abgeschlossen (vgl. Punkt 1.2.2.2), kann mit der Bewertung der gefundenen Konzepte begonnen werden. Ziel der Bewertung ist es, für jedes Konzept zu bestimmen, inwieweit die zuvor definierten Merkmale im Merkmalskatalog erfüllt sind.

Dazu werden für die zu lösenden Fragestellung Merkmalskataloge erstellt. Anschließend wird an Hand der definierten Metriken für jedes Konzept bezüglich aller vorhandenen Merkmalskataloge je ein **Erfüllungswert ermittelt**.

Als Ergebnis der Bewertung von gefundenen Konzepten kann eine **Rangreihenfolge** zwischen den Ansätzen, Technologien und Standards bezüglich ihrer Eignung zur Lösung der Probleme, die sich aus den Fragestellungen der Diplomarbeit ergeben, hergestellt werden. Die Grundlage für die Auswahl von optimalen Konzepten ist damit geschaffen.

1.2.2.4 Auswahl der optimalen Konzepte

Aus dem vorhergehenden Schritt im Untersuchungsablauf, der Bewertung (vergleichenden Analyse) gefundener Konzepte (vgl. Punkt 1.2.2.3), sind für jedes Konzept, jede Technologie, jeden Standard, usw. die Werte des Ausmaßes der Erfüllung bezüglich der Merkmale im Merkmalskatalog bekannt. Damit kann entschieden werden, welche Konzepte als Hilfsmittel für die Erreichung der bereits genannten Ziele in Frage kommen.

Zu diesem Zweck wird für jeden Bewertungsraster (d.h. für eine Menge von Merkmalen, die zur Lösung bestimmter Fragestellungen definiert wurde) das optimale Konzept bestimmt. Das optimale Konzept ist jenes, das bezüglich des Merkmalskatalogs den **höchsten summierten Erfüllungswert** erreicht.

Die Gesamtheit der auf diese Weise bestimmten Ansätze, Technologien und Standards stellt das „Instrumentarium“ zur Realisierung eines Konzepts bzw. einer Architektur für die Gestaltung eines Contentpools zur Verfügung.

1.2.2.5 Konzeptionelles Design der Werkzeuge

Nach dem Abschluss der Bewertung gefundener Konzepte, Ansätze, Technologien und Standards (vgl. Punkt 1.2.2.3) sowie der Auswahl von optimalen Lösungen (vgl. Punkt 1.2.2.4) kann zum Entwerfen eines Konzeptes für die Werkzeuge der Contentpool Verwaltung übergegangen werden. Dies entspricht der Design-Phase im Ablaufmodell des Phasenschemas der Software Entwicklung (vgl. [Oestereich, 1997], S. 63 – 73).

Als Ergebnisse der Projektphase „Design" sind folgende Dokumente zu erstellen:

- **Architektur** des Contentpools und der Werkzeuge zu dessen Verwaltung
- **Tools- und Feature-Katalog** (d.h. Auflistung der zu implementierenden Funktionalität)

Die Architektur des Contentpools bestimmt, wie das **Aufgabenmodell**, das **Datenmodell**, das **Benutzermodell** sowie das **Interaktionsmodell** der zu schaffenden Anwendungen gestaltet werden. Daraus wird der Bedarf für Art und Anzahl der zu entwerfenden Werkzeuge ersichtlich. Wenn für die Werkzeuge zusätzlich die Funktionalität nach Art und Umfang spezifiziert wurde, kann ein Tools- und Feature-Katalog entworfen werden. Im Tools- und Feature-Katalog ist die zu implementierende Funktionalität je Werkzeug (= Tool) aufgelistet.

Um die genannten Dokumente zu erstellen, ist als Input zusätzliche Literatur über die zuvor ausgewählten Konzepte, Ansätze, Technologien und Standards hinzuzuziehen. Dabei ist jene Literatur von Interesse, die sich mit Fragen der Umsetzung und Implementierung von Systemen des Content Managements oder ähnlichen Systemen beschäftigt.

1.2.2.6 Entwerfen der statischen Softwarestrukturen

Auf Grundlage der konzeptionellen Architektur, die im vorhergehenden Schritt entworfen wurde, sind die Strukturen der zu schaffenden Software sowohl aus statischer Sicht als auch aus dynamischer Sicht zu bestimmen. In diesem Abschnitt wird die statische Sicht behandelt, die dynamische Sicht im Abschnitt 1.2.2.7. Zur Beschreibung beider Sichten auf die zu schaffende Software verwendet der Autor objektorientierte Entwurfsmethoden als Hilfswerkzeug.

Die statischen Softwarestrukturen legen fest, welche **Klassen** für die Implementierung der zu schaffenden Werkzeuge benötigt werden. Weiters wird für die zu schaffenden Werkzeuge der **Bedarf an Daten** in ein Modell zusammengefasst. Mit anderen Worten wird das Datenmodell für den Contentpool spezifiziert. Das fertige Datenmodell (aus rein statischer Sicht) sowie das konzeptionelle Klassenmodell der Werkzeuge sind der Output der Projektphase „Entwerfen der statischen Softwarestrukturen".

1.2.2.7 Entwerfen der dynamischen Softwarestrukturen

Neben der Spezifikation beteiligter Klassen und der von ihnen benötigten Daten, also der Beschreibung der Strukturen der zu schaffenden Software aus statischer Sicht (vgl. Punkt 1.2.2.6), ist auch die Betrachtung der dynamischen Abläufe in den zu schaffenden Werkzeugen von Bedeutung. Gegenstand dieser Projektphase ist demnach die Spezifika-

tion der **Interaktion** zwischen den Klassen und den Daten der zu schaffenden Werkzeuge für die Verwaltung eines Contentpools.

Unter Zuhilfenahme von **objektorientierten Entwurfsmethoden** werden einerseits die Interaktion der Werkzeuge untereinander und andererseits die Interaktion zwischen den Werkzeugen und den Benutzern modelliert. Als Ergebnis der Modellierung der Interaktion von Werkzeugen untereinander wird das Klassenmodell verfeinert und durch dynamische Aspekte ergänzt. Als Ergebnis der Modellierung der Interaktion zwischen Werkzeugen und Benutzer entsteht eine Menge von Interaktionsprozessmodellen. Beide Modelle stellen den Output der Projektphase „Entwerfen der dynamischen Softwarestrukturen" dar.

1.2.2.8 Implementierung der Werkzeuge

Nach abgeschlossenem Design der zu schaffenden Werkzeuge (vgl. die Punkte 1.2.2.5 Konzeptionelles Design der Werkzeuge, 1.2.2.6 Entwerfen der statischen Softwarestrukturen und 1.2.2.7 Entwerfen der dynamischen Softwarestrukturen) beginnt die Implementierung der Werkzeuge mit Hilfe der **Programmiersprache Java**. Diese Programmiersprache wurde gewählt, da sie optimale Voraussetzungen für die Nutzung von Internet-Technologien bietet. Die Implementierung in Java war eine Bedingung des Scholion WB+ Projektauftrags (vgl. [Auinger, 2002]).

Als Hilfsmittel für die Implementierung werden Methoden der **objektorientierten Implementierung** verwendet (z.B. ein objektorientierter visueller Editor zum Bearbeiten des Quellcodes). Die zuvor geschaffenen Modelle der Interaktion (Klassenmodell und Interaktionsprozessmodelle) dienen als Input für die Projektphase „Implementierung". Als Output der Projektphase „Implementierung" stehen die zu schaffenden Werkzeuge in einer ersten fertigen, vorerst ungetesteten Version zur Verfügung. Die Werkzeuge bieten folglich die geforderte Funktionalität, ohne noch getestet worden zu sein.

1.2.2.9 Testen der fertig gestellten Werkzeuge

Sobald lauffähige, aber ungetestete Versionen der zu schaffenden Software zur Verfügung stehen (als Output der Projektphase „Implementierung", vgl. Punkt 1.2.2.8), sind diese auf Fehler zu testen. Ziel der Projektphase „Testen" ist es, möglichst viele **Fehler** in den vorläufigen Versionen der Werkzeuge zur Contentpool-Verwaltung **zu finden**. Als Hilfsmittel für die Projektphase „Testen" dienen anerkannte Methoden zum Testen von Software (vgl. z.B. [RechPom, 1999]).

Output der Projektphase „Testen" sind die Testergebnisse. Diese bestehen aus einem Katalog von gefundenen Fehlern (welche zu beheben sind), sowie aus dem gesammelten Feedback der als Tester eingesetzten Benutzer.

1.2.2.10 Abgleich der Werkzeuge mit dem Ausgangsproblem

In der letzten Projektphase dienen die während der Testphase gewonnenen Erkenntnisse (vgl. Punkt 1.2.2.9) als Grundlage für die Erstellung der ersten **Release-Version** der zu schaffenden Werkzeuge. Als Input werden der Fehlerkatalog und das Feedback der Benutzer eingesetzt. Die gefundenen, im Fehlerkatalog dokumentierten Fehler sind zu be-

heben. Von den Test-Benutzern im Feedback angeführte Mängel bezüglich der Benutzbarkeit (Ergonomie) sind ebenfalls zu beheben.

Zusätzlich wird die Software mit den im Ausgangsproblem definierten **Anforderungen** an die Funktionalität verglichen. Fehlen wichtige Anforderungen, so ist die fehlende Funktionalität noch zu implementieren. Nach abgeschlossener letzter Projektphase „Abgleich der Werkzeuge mit dem Ausgangsproblem" steht als Output derselben eine funktionsfähige, der Ausgangs-Spezifikation entsprechende Version der Werkzeuge zur Verwaltung des Contentpools zur Verfügung.

1.3 Hinweise zum Aufbau dieser Arbeit

Um dem Leser den Umgang mit der vorliegenden Diplomarbeit zu erleichtern, gibt dieser Abschnitt Hinweise zur Gestaltung des Textes. Es wird zunächst der Einsatz von verschiedenen Hervorhebungsarten im Text (Formatierungen) erläutert (Abschnitt 1.3.1). Danach gehe ich auf die grundsätzliche Gliederung des Dokuments ein (Abschnitt 1.3.2).

1.3.1 Einsatz von Formatierungen des Textes

Der Autor versuchte bei der Gestaltung der Diplomarbeit, mit Hervorhebungen (verschiedenen Formatierungen des Textes) möglichst sparsam umzugehen. So wurden diese nur eingesetzt, um dem Leser einen Hinweis zur Bedeutung verschiedener Textstellen zu geben. In der Diplomarbeit werden die folgenden Hervorhebungen verwendet:

- **Fettdruck** kennzeichnet wichtige Schlagwörter im Text. Fett gedruckte Begriffe können meist im Schlagwortverzeichnis oder im Glossar der Diplomarbeit nachgeschlagen werden. Weiters wird Fettdruck für die Beschriftung von Abbildungen, Tabellen und ähnlichen Elementen im Text eingesetzt.
- *Kursivdruck* wird meist zur Hervorhebung englischsprachiger Begriffe verwendet, die vom Autor nicht übersetzt wurden. Dies kann aus Gründen des leichteren Verständnisses erfolgen oder weil es in der Fachsprache noch keine weithin anerkannte Übersetzung gibt.
- Referenzen auf verwendete Literatur werden stets in [eckigen Klammern] angegeben. Zu jeder Abkürzung in Klammern findet sich im Literaturverzeichnis (Abschnitt 10.1: Referenzen, Seite 322 ff.) die entsprechende Quelle im Detail angeführt.
- Pseudocode, Code in einer Programmiersprache, Namen von Klassen und Methoden in einem konzeptuellen Klassenmodell oder Code von XML-Schemata und XML-Dokumenten werden in einer `besonderen Druckschrift mit fester Zeichenbreite` wiedergegeben. Diese Vorgehensweise entspricht international gebräuchlichen Konventionen zur Darstellung von Code.

1.3.2 Gliederung des Dokuments

Der grundsätzliche Aufbau der Diplomarbeit ist durch eine „Einkreisung" der behandelten Fragen und Probleme gekennzeichnet. Es wird gezeigt, wie durch systematische Analyse von themenverwandten Konzepten („Related Work") und Zerlegung der gestellten Fragen

und Problemstellungen der untersuchte Problembereich immer weiter eingegrenzt und schließlich analysiert wird. Wo dies möglich war, wird auf bereits gefundene Lösungsansätze unter Wahrung der Grundsätze des wissenschaftlichen Arbeitens zurückgegriffen. Die verbleibenden Probleme versuchte ich so zu lösen, dass ein meiner Meinung nach adäquates und akzeptables Konzept daraus resultiert ist.

Aus diesem Aufbau der Vorgehensweise zur Lösung der Fragen und Problemstellungen ergibt sich die Gliederung in die folgenden Kapitel:

- 1: Einleitung .... Seite 1

  Das vorliegende erste Kapitel führt in die Diplomarbeit ein.

- 2: Ausgangssituation und Problemstellung .... Seite 15

  Im zweiten Kapitel wird die Ausgangssituation (Stand der aktuellen Forschungen und Ähnliches) dargelegt. Unter Berücksichtigung der Ziele dieser Arbeit werden dem Leser die Fragen und Problemstellungen näher gebracht.

- 3: Auswahl der Konzepte und Technologien .... Seite 27

  Das dritte Kapitel wählt die Forschungsansätze und die praktischen Konzepte aus, deren Charakeristika durch eine Analyse von Fachliteratur beleuchtet werden. Das Kapitel bestimmt also jene Konzepte, welche im darauf folgenden Kapitel wissenschaftlich analysiert werden.

- 4: Related Work .... Seite 35

  Hier erfolgt die wissenschaftliche Analyse der zuvor ausgewählten Konzepte aus Forschung und Praxis. Zweck der Analyse ist es, jene Problemfelder zu identifizieren, in denen bereits Lösungen oder Lösungsvorschläge gefunden bzw. entwickelt wurden. Die Untersuchung der themenverwandten Arbeit dient zur Eingrenzung der Problemfelder.

- 5: Bewertung der Konzepte und Technologien .... Seite 123

  Im Kapitel 5 werden aus den untersuchten Konzepten diejenigen Arbeiten ausgewählt, die geeignet sind, auf ihnen die Lösung für die Fragen und Problemstellungen der vorliegenden Arbeit aufzubauen. Die Auswahl der geeigneten Konzepte erfolgt mit Hilfe einer formalen Bewertung an Hand von Merkmalen.

- 6: Design der Software .... Seite 143

  Mit dem Design der Software werden die Antworten gegeben, die nach Meinung des Autors die Fragen und Problemstellungen der Diplomarbeit zufriedenstellend lösen. Im Design werden, soweit dies möglich ist, vorhandene Konzepte aus Forschung und Praxis einfließen. Wo Lücken zu einem kompletten Design der Software bestehen, werden diese durch eigene Lösungen und Entwicklungen geschlossen.

- 7: Implementierung der Software .... Seite 249

  Die Implementierung der Software beschäftigt sich mit der Umsetzung des zuvor entworfenen Software-Designs in einer konkreten Programmiersprache. Für die vorliegende Arbeit wurde die Programmiersprache Java gewählt. Die Gründe für

die Entscheidung „pro Java" können im Abschnitt 1.2.2.8 (Seite 11) nachgelesen werden.

- 8: Test der Software .... Seite 305

  Im Rahmen des Projekts Scholion WB+ wurden nach Fertigstellung der Software funktionale Tests durchgeführt. Das achte Kapitel präsentiert Erfahrungen und Rückmeldungen, die während der Testphase gewonnen werden konnten. Somit werden empirische Erkenntnisse zum Einsatz und zur Akzeptanz der Software präsentiert.

- 9: Fazit und Ausblick .... Seite 314

  Den Abschluss der Ausführungen in dieser Arbeit bildet im neunten Kapitel eine Darstellung von Zielen, die erreicht werden konnten, und solchen, die erst noch erreicht werden müssen. An Hand der im Abschnitt 1.1 (Ziele dieser Diplomarbeit, siehe Seite 2 ff.) präsentierten und im Kapitel 2 präzisierten Ziele zieht der Autor Bilanz über die Tätigkeiten, die im Rahmen des Projekts Scholion WB+ durchgeführt wurden, und wie diese zum Erreichen der Ziele der Diplomarbeit beitragen konnten.

- 10: Anhang .... Seite 322

  Im Anhang findet der Leser weitere Unterstützung für die Verwendung dieser Diplomarbeit. Ein Glossar führt in die wichtigsten Begriffe dieser Arbeit ein, wobei trotzdem Vorkenntnisse zu bestimmten Themengebieten (z.B. Auszeichnungssprachen, vor allem XML; Programmiersprachen, vor allem Java; Methoden des objektorientierten Entwurfs und der objektorientierten Programmierung; usw.) vorausgesetzt werden.

Hinweis: Im gesamten Dokument werden zur Bezeichnung von Gliederungseinheiten des Textes die Begriffe Kapitel, Abschnitt und Punkt verwendet. Diese Begriffe besitzen die folgende Bedeutung:

- Ein *Kapitel* bezeichnet einen Textabschnitt der obersten Gliederungsebene. Die Diplomarbeit ist in 10 Kapitel gegliedert, die weiter oben kurz umrissen wurden.
- Ein *Abschnitt* bezeichnet einen Textabschnitt der zweiten oder dritten Gliederungsebene. Folglich besteht ein Kapitel aus mehreren Abschnitten. Ein Abschnitt der Gliederungsebene 2 kann jeweils aus weiteren (Unter-) Abschnitten bestehen.
- Wenn eine noch präzisere Gliederung erforderlich ist, wird von *Punkten* gesprochen. Ein Punkt bezeichnet also einen Textabschnitt der vierten Gliederungsebene oder tiefer. Wo dies möglich ist, wird eine allzu tiefe Gliederung des Dokuments jedoch vermieden.

Der Autor hat versucht, durch eine große Anzahl von **Querverweisen** im Dokument dem Leser die Navigation im Text so leicht wie möglich zu machen. So findet sich zu Beginn jedes Kapitels und jedes Abschnittes eine Übersicht der Inhalte, die den Leser auf den nachfolgenden Seiten erwarten. Im Fließtext wurden Querverweise auf interessante Textstellen anderswo im Dokument jedesmal dann eingefügt, wenn aus meiner Sicht damit die Verständlichkeit und/oder die Lesbarkeit erhöht werden konnten.

2 Ausgangssituation und Problemstellung

Ein wesentlicher Schwerpunkt in den Forschungsaktivitäten am Institut für Wirtschaftsinformatik, Communications Engineering[3] (Johannes Kepler Universität Linz) liegt seit Jahren auf der Weiterentwicklung von **Wissenstransfer-Konzepten**. Dazu gehört die Entwicklung ausreichender Unterstützung von Advanced Distance Learning (ADL) durch entsprechende Werkzeuge.

Zu diesem Zweck wurde im Jahr 1998 eine Wissenstransfer-Umgebung namens SCHOLION konzeptuell entwickelt und in einen Prototypen umgesetzt ([Ellmer et. al., 1998], [Froschauer et al., 1999]). Das österreichische Ministerium für Wissenschaft und Verkehr finanzierte ein Nachfolgeprojekt für die Weiterentwicklung dieses Projektes (GZ50004/2-III/9/99, bmwv). Von Dr. Andreas Auinger und Mag. Wolfgang Karrer wurde der Prototyp analysiert und zu einer interaktiven Multimedia-Anwendung zur computerbasierten Unterstützung von Wissenstransfer („Teleteaching und Telelearning") weiterentwickelt ([Auinger, 1999], [Karrer, 2000]).

In einem weiteren, im Juni 2002 gestarteten **Nachfolgeprojekt** bestand das Ziel darin, die von den genannten Personen begonnene Arbeit fortzusetzen. Dabei wurde großes Augenmerk darauf gelegt, die in den letzten Jahren durch die rasanten technischen Fortschritte und die weitgehende Verbreitung des Internets entstandenen Möglichkeiten zu erschließen. Unter Nutzbarmachung der neuen **Internet-Technologien** besteht das Ziel darin, die Benutzbarkeit und Mächtigkeit der Wissenstransfer-Software zu verbessern.

Auf Grundlage der **Java Servlet Technologie** (vgl. [Sun, 1999]) und der Auszeichnungssprache XML (vgl. [W3C, 2000]) wird der Scholion Software eine völlig veränderte Architektur zu Grunde gelegt, wodurch bedeutende Fortschritte bei der **Geschwindigkeit** und **Robustheit** der Software zu erwarten sind. Teile der Ergebnisse der Arbeiten an der Neuentwicklung werden in diesem Dokument vorgestellt. (Für die Darlegung der übrigen Ergebnisse wird der Leser auf [Fenneberg, 2002], [Fürlinger, 2003], [Radmayr, 2003] und [Mielach, 2003] verwiesen.)

Hinweis: Die Programmiersprache Java wurde für die Implementierung der neuen Scholion Software gewählt, da sie optimale Voraussetzungen für die Nutzung internetbasierter Technologien bietet. Diese Entscheidung wurde bereits vor Vergabe des Auftrages für das Reengineering-Projekt getroffen und war eine **Bedingung im Projektauftrag** [Auinger, 2002]. Es würde den Rahmen dieses Dokuments sprengen, sich mit alternativen Programmiersprachen zu beschäftigen oder eine einschlägige Evaluierung durchzuführen.

In heute gängigen und auf dem Markt erhältlichen virtuellen, verteilten Lernumgebungen sind **elektronische Lernmaterialien** schon üblich und eine Möglichkeit zum Durcharbeiten der Materialien in die Lernumgebung integriert (vgl. dazu [Reindl, 2001]). Bisher fehlt aber die Möglichkeit, Lernmaterialien zusammen zu stellen, die ganz auf die **individuellen Vorkenntnisse, Bedürfnisse und Vorlieben** eines Benutzers abgestimmt sind. Als Idealvorstellung sollte eine Lernumgebung einen Bestand an gespeichertem

[3] Siehe http://www.ce.jku.at

Wissen (ein „*Repository*" in Form elektronischer Materialien) zu verschiedenen Themen bieten. Die Benutzer des Systems können sich nach ihren individuellen Vorstellungen zu einem beliebigen Thema ihre eigenen Unterlagen erstellen, wobei als Quelle das *Repository* dient.

Diesem Gedankengang folgend wurde die **Projektidee** geschaffen, einen so genannten Contentpool zu entwickeln. Das Ziel der Erstellung eines Contentpools ist, wie oben erwähnt, den Benutzern eine Menge an **elektronischen Inhalten** zur Verfügung zu stellen, die für die weitere Verwendung in elektronischen Lernmaterialien bereits vorbereitet wurden. Im Contentpool werden also elektronische **Dokumente in zerlegter Form** abgelegt, wobei Information über die **Semantik der Inhalte** (also über deren Bedeutung bzw. das von ihnen behandelte Thema) in den **Metadaten** der Inhalte mitgespeichert wird.

Aus den gespeicherten Dokumenten und Dokument-Teilen im Contentpool können sich Benutzer **Kursmaterialien und Nachschlagewerke** (= Lernmaterialien) nach individuellen Bedürfnissen und Vorlieben erstellen. Die zur semantischen Bedeutung der Dokumente gespeicherten Metadaten werden bei der Zusammensetzung eines Kursmaterials / Nachschlagewerks mit berücksichtigt.

Diese Vorgaben erlauben die Definition der globalen Fragestellung dieser Diplomarbeit. Aus der globalen Fragestellung werden durch Zerlegung weitere, detaillierte Fragestellungen aufgeworfen und präzisiert. Die Fragestellungen geben den weiteren Verlauf der in dieser Diplomarbeit präsentierten Untersuchung vor.

2.1 Globale Fragestellung

Aus den Zielen für das Projekt Scholion WB+ und, daraus folgend, den Zielen dieser Diplomarbeit (vgl. dazu Abschnitt 1.1, Seite 2) lässt sich die globale Fragestellung ableiten:

> *Wie müssen Lehrmaterialien zerlegt werden, um unabhängig von ihrem ursprünglichen Kontext in einem neuen Zusammenhang dargestellt oder zitiert werden zu können? Wie kann die Speicherung zerlegter Lehrmaterialien optimal erfolgen, so dass dabei die semantischen Zusammenhänge nicht verloren gehen?*

Die wesentlichen Schlagwörter und folglich die sich daraus ergebenden Fragestellungen sind folgende:

- **Zerlegung**: Wie können digitalisierte Lehrmaterialien zerlegt werden? Nach welchen Kriterien soll die Aufteilung eines Dokuments erfolgen?
- **Berücksichtigung der Semantik**: Wie kann sichergestellt werden, dass Wissensatome (zerlegte Lehrmaterialien), die inhaltlich miteinander in Zusammenhang stehen, beim Erstellen von Kursmaterialien oder Nachschlagewerken zum fraglichen Inhalt tatsächlich alle berücksichtigt werden? Welches Datenmodell ist geeignet, um die semantischen Zusammenhänge zwischen Wissensatomen abzubilden?

- **Speicherung von Wissensatomen**: Welche Möglichkeiten stehen zur Verfügung, um sowohl Wissensatome (zerlegte Dokumente) als auch semantische Zusammenhänge (d.h. semantische Beziehungen) speichern zu können? Wie kann die Speicherung optimiert werden, so dass der Bedarf an Speicherplatz bzw. die Zugriffszeit minimal bleiben?
- **Darstellung der Wissensatome**: Mit welchen Mitteln kann die Verwaltung eines Contentpools implementiert werden? Wie können Wissensatome, wie die semantischen Zusammenhänge verständlich dargestellt werden? Welche Elemente bzw. Darstellungmittel stehen zur Verfügung, um eine geeignete Benutzungsschnittstelle zu designen? Wie muss die Darstellung aussehen, um ein Höchstmaß an Benutzbarkeit, Ergonomie und Effizienz zu bieten?
- **Verwendung der Wissensatome**: Wie können Wissensatome in Lernmaterialien und Nachschlagewerke eingebaut werden? Wie kann man bei der Zusammenstellung eines Lernmaterials oder Nachschlagewerk sicherstellen, dass alle zu einem vorgegebenen Thema relevanten Wissensatome berücksichtigt werden?

Die hier nur kurz angedeuteten Fragen werden auf den kommenden Seiten ausführlich diskutiert.

2.2 Detailierung der Problemstellung

Aus der globalen Fragestellung der Diplomarbeit ergeben sich, wie im Abschnitt 2.1 im Überblick dargestellt, durch Zerlegung weitere Probleme und Fragen:

- Zerlegung von Dokumenten (Abschnitt 2.2.1)
- Aufbau eines Contentpools (Abschnitt 2.2.2)
- Anwendung des Wissens aus dem Contentpool (Abschnitt 2.2.3)

In der Folge werden diese Fragestellungen präzisiert.

2.2.1 Zerlegung von Dokumenten

Die Zerlegung elektronischer Dokumente (z.B. Lehrmaterialien, elektronische Papers, digitalisierte Bücher, usw.) dient zum Aufbau des *Repository* aus zerlegten elektronischen Büchern und sonstigen Materialien. Der **Bestand an Wissensatomen** wird auf diese Weise aufgebaut und erweitert. Beim Zerlegen der Dokumente ist darauf zu achten, den **semantischen Zusammenhang** des Inhalts eines Wissensatoms nicht zu verlieren, sondern ihn im semantischen Netz des Contentpools mitzuspeichern.

Folgende Punkte müssen bei der Zerlegung von Dokumenten folglich beachtet werden:

- Die Speicherung des ursprünglichen semantischen Zusammenhangs muss in einer geeigneten Form (z.B. Beschlagwortung) erfolgen.
- Die **Definition eines Wissensatoms**, d.h. Spezifikation der Daten, die als Wissensatom und zusammen mit einem Wissensatom abgespeichert werden sollen, hat zu erfolgen.

- Die Frage, wie **Wissensatome generiert** werden können und sollen (automatisiert, d.h. mit Hilfe eines Algorithmus, oder durch einen Autor), muss beantwortet werden. Jene der Möglichkeiten, die größere Effizienz beim Aufbau des Contentpools gewährleistet, ist zu bevorzugen.

2.2.2 Aufbau eines Contentpools

Um einen Contentpool aufbauen zu können, müssen die Probleme der **Speicherung von Wissensatomen** gelöst werden. Daraus begründet sich die Aufgabe, ein **Datenmodell** für den Contentpool zu entwerfen. Ein geeignetes Datenmodell des Contentpools muss in der Lage sein, sowohl die Daten selbst (d.h. die Wissensatome als Teile von elektronischen Lehrmaterialien), als auch Daten über die semantischen Inhalte und Zusammenhänge zu speichern.

Entsprechend der Aufteilung des Contentpools in die beiden Komponenten **Datenbasis** und **semantisches Netz** ist auch das Datenmodell in zwei Teilen zu definieren:

- Datenmodell für die Datenbasis
- Datenmodell für das semantische Netz

2.2.3 Anwendung des Wissens aus dem Contentpool

Als letzter Schritt zum Aufbau eines Contentpools muss gewährleistet sein, dass die in der Datenbasis des Contentpools gespeicherten Inhalte effektiv und effizient **genutzt** werden können. Es sind folglich Probleme der **Anwendung von Wissensatomen** zu lösen.

Die Anwendung von Wissensatomen wirft folgende Problemstellungen auf:

- Der **Zugriff** auf die gespeicherten Wissensatome muss möglichst effizient realisiert werden. Eine gezielte **Suche nach semantischen Kriterien** ist ebenso zu unterstützen wie ein beliebiges, vom Benutzer definiertes Navigieren der Wissensatome (ein „Browsen" in den Inhalten).
- Die oben genannten Ansprüche können nur erfüllt werden, wenn ein geeignetes Mittel zur **Darstellung von Wissensatomen** gefunden wird. Die zur Darstellung eines Wissensatoms gedachte Benutzungsschnittstelle muss in der Lage sein, sämtliche zum Wissensatom gespeicherte Information direkt oder indirekt zum Benutzer zu transportieren.
- Schließlich ist die Erstellung von benutzerdefinierten Kursmaterialien und Nachschlagewerken durch ein **Werkzeug** zu unterstützen.

2.3 Ziele für das Design der Software

Die dargelegten Problemstellungen geben die Ziele für das **konzeptuelle Design** der zu schaffenden Werkzeuge vor. Inhalte in Form von Wissensatomen müssen aufbereitet, gespeichert und verwaltet werden können. Auf die Datenbasis aus Wissensatomen muss ein effizienter Zugriff zum Zweck der Erstellung neuer Dokumente gewährleistet sein. Die Dokumente müssen auf die Bedürfnisse und Vorkenntnisse einer individuellen Person

abgestimmt werden können. Der Umgang mit dem Contentpool muss für alle Benutzer so einfach wie möglich gestaltet sein. Daraus lassen sich folgende Hauptkategorien von Zielen bilden.

- 2.3.1 Allgemeine Benutzbarkeitskriterien
- 2.3.2 Nutzung von Content
- 2.3.3 Speicherung von Content
- 2.3.4 Aufbereitung von Content

2.3.1 Allgemeine Benutzbarkeitskriterien

Je besser und einfacher ein Softwaresystem zu benutzen ist, desto eher profitieren seine Benutzer vom Einsatz des Systems. Desto höher wird zudem die Akzeptanz durch die Benutzer ausfallen.

Wie für alle Softwaresysteme üblich, müssen solche **ergonomischen Gesichtspunkte** auch bei der Erstellung des Contentpools für Scholion WB+ berücksichtigt werden. Stary gibt Kriterien an, nach denen interaktive Systeme gestaltet sein sollen (vgl. [Stary, 1996]). Scholion WB+ ist ein **interaktives System**, d.h. es ist ein **Dialogbetrieb** zwischen System und Benutzer vorgesehen. Daher sind die von [Stary, 1996] angegebenen Kriterien auch auf das Design von Scholion WB+ anwendbar. Für die Benutzbarkeit (auch Benutzerfreundlichkeit genannt) des Contentpools ergeben sich also folgende Ziele:

- Einfachheit der Schnittstelle (Aufgabenangemessenheit)
- Selbstbeschreibungsfähigkeit der Benutzungsschnittstelle
- Steuerbarkeit (leichte Bedienbarkeit)
- Fehlertoleranz und Robustheit (Erwartungskonformität)
- Aktualität und Richtigkeit der Information
- Nachvollziehbarkeit der Information
- Erlernbarkeit
- Rasche Antwortzeiten bei der Bearbeitung von Aufgaben

Die genannten Ziele werden nachfolgend detailliert erläutert.

2.3.1.1 Einfachheit der Schnittstelle (Aufgabenangemessenheit)

Der Benutzer darf nicht mit den Eigenschaften der Interaktionshilfsmittel unnötig belastet werden. An der grafischen Benutzungsschnittstelle sollte **genau jene Information** präsentiert werden, die zur Bewältigung der Aufgabe benötigt wird, nicht mehr und nicht weniger. Darüber hinaus dürfen beim Benutzer keine Programmierkenntnisse vorausgesetzt werden [Stary, 1996].

2.3.1.2 Selbstbeschreibungsfähigkeit der Benutzungsschnittstelle

Selbstbeschreibungsfähigkeit einer (grafischen) Benutzungsschnittstelle ist gegeben, wenn einer der folgenden zwei Aspekte erfüllt wird [Stary, 1996]:

- Jeder einzelne Interaktionsschritt ist entweder **unmittelbar verständlich**, oder
- auf Wunsch wird den Benutzern eine **Erläuterung** des Einsatzzweckes und des Leistungsumfangs der Schnittstelle angeboten (so genannte Online- oder Echtzeit-Hilfe).

2.3.1.3 Steuerbarkeit (leichte Bedienbarkeit)

Steuerbar ist ein System dann, wenn Benutzer die **Geschwindigkeit** der Interaktion sowie die **Auswahl und Reihenfolge** der Interaktionshilfsmittel beeinflussen können. Darüber hinaus ist es wünschenswert, dass Benutzer die Art und den **Umfang** von Ein- und Ausgaben beeinflussen können [Stary, 1996].

2.3.1.4 Fehlertoleranz und Robustheit (Erwartungskonformität)

Ein Benutzer stellt stets Erwartungen an eine von ihm verwendete Software (vgl. [Stary, 1996]). Erwartungskonformität ist gegeben, wenn

- ... die Erwartungen der Benutzer erfüllt werden. Erwartungen entstehen aus **Erfahrungen**, die Benutzer aus Arbeitsabläufen aufgrund der Transparenz und Konsistenz des Systems während des Umgangs mit der Benutzungsschnittstelle sammeln können.
- ... die **Konzeption der Arbeitsabläufe** menschlich durchschaubar ist (**Transparenz**), sowohl die Aufgaben, als auch die Interaktionshilfsmittel betreffend.
- ... der Aufbau der Benutzungsschnittstelle regelhaft erfolgt (**Konsistenz**), d.h. dass der Aufbau sowie der Ablauf von Interaktionen in gleichartigen Situationen durch jeweils **ähnliches Systemverhalten** gekennzeichnet ist.

Zum Aspekt „Erwartungskonformität" müssen auch Fehler gerechnet werden, die während der Interaktion des Systems mit dem Benutzer auftreten. Fehler sind niemals mit den Erwartungen der Benutzer konform. Ein System ist **fehlertolerant bzw. fehlerrobust**, wenn trotz erkennbarer Falscheingaben das angestrebte Arbeitsergebnis mit **minimalem Korrekturaufwand** erreicht werden kann. Das bedeutet vor allem [Stary, 1996]:

- Fehler müssen dem Benutzer **verständlich** gemacht werden.
- Eine Falscheingabe darf nicht zu einem **undefinierten Systemzustand** oder gar zu einem **Zusammenbruch** des Systems führen.
- Fehlermeldungen müssen **sofort ausgegeben** werden (unmittelbares Feedback).

2.3.1.5 Aktualität und Richtigkeit der Information

An einer Benutzungsschnittstelle darf nur jene Information präsentiert werden, welche **noch in dieser Form gültig** ist. Aktualität und Richtigkeit der Information wären ver-

letzt, wenn dem Benutzer zum Beispiel das Vorhandensein von Wissensatomen angezeigt wird, die bereits aus dem Contentpool gelöscht worden sind. Ein solcher Zustand muss von der Software vermieden werden (vgl. [Stary, 1996]).

Aktualität und Richtigkeit betrifft auch den in den Wissensatomen gespeicherten Inhalt. Es muss vermieden werden, dass Inhalte, die in der Datenbasis des Contentpools gespeichert werden, bereits durch neue wissenschaftliche Erkenntnisse überholt und damit nicht mehr aktuell sind. Benutzer müssen sich darauf verlassen können, dass das im Scholion WB+ Contentpool angebotene Wissen dem **neuesten Stand der Forschungen** entspricht.

2.3.1.6 Nachvollziehbarkeit der Information

Eng verwoben mit der Forderung nach Aktualität und Richtigkeit ist das Ziel, die im Contentpool gespeicherte Information so zu präsentieren, dass Inhalte ihrem **Urheber zugeordnet** werden können [Stary, 1996]. Dieser Aspekt ist aus folgenden beiden Gründen wichtig:

- Interindividuelle Nachvollziehbarkeit ist ein wesentliches Kriterium bei der Erstellung wissenschaftlicher Dokumente. Die in Scholion WB+ zu erstellenden Kursmaterialien und Nachschlagewerke müssen den allgemeinen Wissenschaftlichkeitskriterien genügen.
- Auf diese Weise kann auch die Einhaltung von **urheberrechtlichen Bestimmungen** gewährleistet werden. So ist es z.B. denkbar, dass ein Autor die Veränderung der Reihenfolge in den Texten seiner Publikationen untersagt, oder die Veröffentlichung seiner Publikationen mit denen bestimmter anderer Autoren, usw. Im Hinblick auf eine angestrebte Kommerzialisierung des Systems kommt dem Argument eine besondere Wichtigkeit zu.

2.3.1.7 Erlernbarkeit

Die Bedienung einer Benutzungsschnittstelle ist dann erlernbar, wenn es für einen Benutzer möglich ist, die Bewältigung von Aufgaben in einer **angemessenen Zeitspanne** zu erlernen. Zu diesem Zweck ist es empfehlenswert, dass der Lernprozess von Seiten des Programmsystems aktiv unterstützt wird, z.B. durch eine Tutoring-Komponente [Stary, 1996].

2.3.1.8 Rasche Antwortzeiten bei der Bearbeitung von Aufgaben

Besonders wichtig für die Akzeptanz seitens der Benutzer ist die Möglichkeit, die anfallenden Aufgaben rasch zu bewältigen [Stary, 1996]. Im Kontext der Erstellung von Wissensatomen bzw. der Pflege der Datenbasis des Contentpools wird dieses Ziel zum Teil nicht leicht zu erreichen sein.

Eine genauere Festlegung der zu erreichenden Antwortzeiten ist schwer möglich, da sich Lernmaterialien nach Art des Inhalts und Umfang voneinander wesentlich unterscheiden können. Für das Design des Scholion WB+ Contentpools ist daher nur die Festlegung der folgenden **Richtlinien** möglich:

- Die **Zerlegung** von Lernmaterialien und die **Aufbereitung** der Wissensatome mit Metadaten müssen, soweit dies durch einen Algorithmus lösbar ist, **automatisiert** geschehen. Auf diese Weise kann die zur Bearbeitung (dazu zählen die Zerlegung in semantische Einheiten, die Aufbereitung der semantischen Einheiten mit Metadaten sowie die Speicherung der semantischen Einheiten als Wissensatome in der Datenbasis des Contentpools) eines elektronischen Dokuments notwendige Zeitspanne wesentlich beeinflusst werden.
- Das Datenmodell muss mit Bedacht auf die möglichst effektive Übertragung von Daten entworfen werden. Bei Transaktionen mit der Datenbasis oder dem semantischen Netz des Contentpools ist sicherzustellen, dass jeweils die **minimale Menge an Daten** übertragen wird.
- Für die Speicherung von persönlichen Dokumenten muss es folglich eine Begrenzung der **zulässigen Dateigröße** geben. Bei sehr großen Dateien (im Umfang von mehreren zehn oder hundert Megabyte), die für die Verwendung durch mehrere Benutzer freigegeben wurden, sind Benutzer mit einer konventionellen Anbindung an das Internet (z.B. mit Hilfe eines Modems) wesentlich benachteiligt, da der Zugang zu diesen Dokumenten innerhalb einer zumutbaren Zeitspanne nicht möglich ist. Das tatsächliche Ausmaß der zulässigen Dateigröße ist im Rahmen der Designphase noch zu quantifizieren.

2.3.2 Nutzung von Content

Letztlich dient der Contentpool in Scholion WB+ dem Zweck, personalisierte Dokumente zu erstellen, die als Lernunterlagen in einem Online-Kurs oder auch für persönliche Lern- und Nachschlagezwecke eingesetzt werden können. Unter Nutzung versteht man folglich **Suche** im Contentpool, **Zusammenstellung** von Wissensatomen und **Exportieren** von Teilen des Contentpools (also von Kursmaterialien und Nachschlagewerken) sowie die **Wartung** der gespeicherten Wissensatome.

Bei der Nutzung des Contents, d.h. beim Zugriff auf den gespeicherten Content, muss zudem eine Differenzierung nach der Rolle des Benutzers vorgenommen, d.h. zwischen der **Nutzung durch Lernende** und der **Nutzung durch Lehrende** unterschieden werden. Es werden nachfolgend die Ziele präsentiert, die für jede der beiden Benutzergruppen relevant sind.

2.3.2.1 Nutzung durch Lernende

Lernende werden lediglich einen eingeschränkten Funktionsumfang des Scholion WB+ Programmsystems, so auch des Contentpools, nutzen können. Für diese Gruppe stehen Ziele im Vordergrund, die die effektive und effiziente Nutzung **bereits vorhandener** Lernmaterialien (das sind Kursmaterialien und Nachschlagewerke) sowie die persönliche Annotation und Gestaltung der Kursmaterialien betreffen. Folglich können Lernende die Suche, das Exportieren und eingeschränkt die Zusammenstellung von Wissensatomen benützen. Konkret ergeben sich folgende Anforderungen:

- Möglichkeit der **Suche von Wissensatomen** nach verschiedenen Bedingungen: insbesondere ist dabei die Suche nach **semantischen Beziehungen** zu berück-

sichtigen, d.h. z.B. Suche nach relevanten Wissensatomen oder die Suche nach Wissen, das auf der aktuellen Menge von ausgewählten Wissenseinheiten aufbaut.

- Bereitstellung einer Schnittstelle zur **Aufbereitung** von Kursmaterialien und Nachschlagewerken mit persönlichen Dokumenten oder multicodalen Elementen
- Erstellung von **Annotationen** innerhalb eines Kursmaterials
- Individuelle **Formatierung** ausgewählter Textstellen
- Auswahl der **gewünschten Codalität** eines Kursmaterials
- Bereitstellung eines oder mehrerer **Exportfilter** zur Abspeicherung von Dokumenten in einem vom Scholion WB+ Programmsystem unabhängigen Format, z.B. pdf

2.3.2.2 Nutzung durch Lehrende

Die unter Punkt 2.3.2.1 angeführten Nutzungsziele gelten auch für die Lehrenden. Ihnen müssen darüber hinaus Werkzeuge zur Verfügung gestellt werden, die nicht nur die Arbeit mit existierenden Lernmaterialien, sondern auch die Erstellung **neuer Lernmaterialien** erlauben. Zudem sind Lehrende jene Benutzer des Systems, denen die **Wartung** der Wissensatome übertragen ist. So ergeben sich zusätzliche Ziele:

- Bereitstellung eines **Werkzeugs zum Erstellen** von Kursmaterialien und Nachschlagewerken
- Funktionalität zum Erstellen von **semantischen Beziehungen** zu bereits abgespeicherten Wissenseinheiten
- Ermöglichung der **Wiederherstellbarkeit** der Originaldokumente (in die ursprüngliche Form)
- Bereitstellung eines Werkzeugs zum Updaten oder Löschen von Wissensatomen, so dass die Datenbasis des Contentpools ständig auf dem **neuesten Stand** der Wissenschaft gehalten werden kann. Veraltetes Wissen kann so aktualisiert oder gelöscht werden.

2.3.3 Speicherung von Content

Die Speicherung stellt die Voraussetzung für die Nutzung des Contents dar. Aus den Zielen, die bei der Nutzung von Content verfolgt werden, lassen sich daher die Ziele bei der Speicherung des Contents ableiten. Von besonderer Wichtigkeit ist die Möglichkeit, ein **semantisches Netz** aus Aussagen über ein Fachgebiet unabhängig von spezifischen Dokumenten aufzubauen. D.h., die semantischen Beziehungen müssen unabhängig von den zerlegten Dokumenten abgespeichert sein.

Im vorliegenden Zusammenhang muss jedoch beachtet werden, dass eine Unterteilung der Ziele nach der Rolle der Benutzer keinen Sinn hat, da die Speicherung der Daten für keinen Benutzer transparent ist, sondern im **Hintergrund von der Verarbeitungslogik** erledigt wird. Alle in Abschnitt 2.3.1 (Allgemeine Benutzbarkeitskriterien) angegebenen Ziele müssen ebenfalls unterstützt werden.

Aus diesen Prämissen ergeben sich folgende Ziele für die Speicherung:

- **Getrennte Speicherung** von semantischen Beziehungen (semantisches Netz) und den eigentlichen Ressourcen (Dokumenten bzw. Dokumentteilen)
- Reservierung eines eigenen Teils der Datenbank zur **Abspeicherung** persönlicher Inhalte
- Bereitstellung einer **Schnittstelle zur Abspeicherung** von persönlichen Inhalten
- Speicherung von **Metadaten** zu den Wissensatomen, die einen Suchvorgang nach verschiedenen Suchkriterien und die Wiederherstellung der gespeicherten Dokumente in der ursprünglichen Form ermöglichen
- Bereitstellung einer **Schnittstelle zur Suche** innerhalb des Contentpools
- Bereitstellung einer Schnittstelle zur **Pflege** der Datenbasis, d.h. zum Löschen veralteter, Aktualisieren überholter und Hinzufügen fehlender Wissensatome im Contentpool
- Entwicklung von Möglichkeiten zur **Versionierung** von Wissensatomen
- Einteilung der gespeicherten Wissensatome in **Kategorien** (so genannte „Kategorisierung"), z.B. Definitionen, Formeln, Abbildungen, ...
- Bereitstellung einer Schnittstelle zur **multicodalen Aufbereitung** der Wissensatome, d.h. zur Konvertierung der Kursmaterialien in verschiedene Darstellungsformen (Codalitäten, z.B. reiner Text, Audiodatei, usw.)

2.3.4 Aufbereitung von Content

Unter der Aufbereitung wird die **Anreicherung** von Inhalten aus Lehrmaterialien mit **Metadaten** verstanden. Metadaten sind Daten, die nähere Information nicht nur zum Inhalt selbst, sondern vor allem auch zum Zusammenhang, in welchem die Inhalte wertvoll sind, enthalten. Dieser Aspekt der Nutzung des Contentpools betrifft lediglich die Rolle der Lehrenden. Für Lernende ist bis auf die Möglichkeit zur Erstellung persönlicher Annotationen keine Editierfunktionalität vorgesehen. Folgende Ziele sind für die Aufbereitung des Contents zu beachten:

- Automatisches **Zerlegen** von digitalen Dokumenten so weitgehend wie möglich
- Automatisches **Aufbereiten** von Wissensatomen mit Metadaten so weitgehend wie möglich
- Bereitstellung einer Schnittstelle zum manuellen **Nachbearbeiten der Zerlegungs-Ergebnisse** eines digitalen Dokuments
- Bereitstellung einer Schnittstelle zum manuellen **Nachbearbeiten und Ergänzen der Metadaten**
- Automatisches Auffinden von **semantischen Beziehungen** zwischen Wissensatomen so weitgehend wie möglich
- Bereitstellung einer Schnittstelle zum manuellen **Nachbearbeiten und Ergänzen** der semantischen Beziehungen

- Sicherstellung einer konsistenten Begriffswelt sowohl innerhalb der Datenbasis als auch innerhalb des semantischen Netzes durch die Bereitstellung einer zentralen „Schlagwortdatenbank", also eines so genannten **Thesaurus**

2.3.5 Sonstige Designziele

Die Erfüllung aller zuvor genannten Ziele hat so zu erfolgen, dass weitere Rahmenziele bei der Entwicklung des Contentpools erfüllt werden. Die weiteren Rahmenziele für das Projekt lauten wie folgt (vgl. [Auinger, 2002]):

- **Plattform-Unabhängigkeit**; der Benutzer wird nicht zur Verwendung eines bestimmten Betriebssystems „gezwungen".
- Unabhängigkeit von **zusätzlicher Software**; die Wissenstransfer-Umgebung Scholion WB+ (folglich auch der Contentpool) sind so zu entwerfen, dass sie in einem **Standard-Webbrowser** ausgeführt werden können.
- **Kompatibilität** zu anderen Wissenstransfer-Umgebungen (**Interoperabilität**), so dass Inhalte nach Möglichkeit von einer in die andere Umgebung transferiert werden können.
- **Offenheit** in den Schnittstellen und der Architektur.
- Hohe **Skalierbarkeit** und Leistungsfähigkeit.

2.4 Zusammenfassung

Im Kapitel 2 wurde zunächst die Ausgangssituation definiert und der Stand der aktuellen Erkenntnis dargestellt. Die durch die Ziele der Diplomarbeit aufgeworfene Problemstellung wurde definiert und durch die Zerlegung in Teilprobleme weiter präzisiert.

Es wurden aus der globalen Problem- bzw. Fragestellung die wichtigsten Fragen bestimmt, die durch die Erkenntnisse der Diplomarbeit zu beantworten sind. Aus den Zielen (vgl. Abschnitt 1.1, Seite 2 ff.) ergibt sich, dass sowohl die Entwicklung eines Contentpool-Konzepts als auch Entwurf und Implementierung von Werkzeugen zum Zugriff auf die Inhalte im Contentpool Gegenstand der Diplomarbeit sind.

Für die Entwicklung des Contentpools wurden folgende Problemstellungen definiert:

- Zerlegung von Dokumenten
- Berücksichtigung der semantischen Zusammenhänge
- Darstellung von Wissensatomen
- Verwendung von Wissensatomen (d.h. Darstellung bzw. Einbauen in neue Dokumente)

Als Ziele für Entwurf und Implementierung von Werkzeugen wurden gesetzt:

- Allgemeine Benutzbarkeitskriterien
- Unterstützung der Nutzung von Wissensatomen
- Unterstützung der Speicherung von Inhalten

- Unterstützung der Aufbereitung von Wissensatomen
- Sonstige Designziele (Rahmenziele)

3 Auswahl der Konzepte und Technologien

Ziel der Auswahl von Konzepten ist das Identifizieren von Konzepten, Ansätzen, Technologien und Standards, die geeignet *scheinen*, die im Kapitel 2 (Ausgangssituation und Problemstellung) genannten Frage- und Problemstellungen zu lösen. (**Hinweis**: ob die gefundenen Konzepte tatsächlich zur Lösung der gestellten Probleme geeignet sind, wird im Rahmen einer Bewertung überprüft. Vgl. dazu Kapitel 5, Seite 123 ff.)

Bevor ich auf die für die Diplomarbeit relevanten Konzepte näher eingehe, erfolgt zunächst eine **Gesamtübersicht** aller Konzepte, Ansätze, Technologien, Standards, usw., die bei der Erstellung der Diplomarbeit berücksichtigt wurden (Abschnitt 3.1). Anschließend wird aus den Frage- und Problemstellungen (vgl. Kapitel 2) sowie dem *state of the art* in der Wissenschaftsdisziplin „E-Learning" eine **Einteilung** der Konzepte **nach Kategorien** vorgenommen (Abschnitt 3.2). Die Kategorien bilden den **Leitfaden** für die Konzeptauswahl sowie für die Literatur- und Konzeptanalyse in Kapitel 4.

Nach der damit erfolgten Einteilung des theoretischen Hintergrunds werden für jede der definierten Kategorien die Konzepte, Standards und Technologien aufgezählt, die aus Sicht des Autors als geeignet für die Lösung der jeweiligen Probleme schienen.

Die **Auswahl** der tatsächlich analysierten Konzepte geschah mit Hilfe von K.O.-Kriterien. Die detaillierte Analyse der ausgewählten Konzepte folgte später und wird im Kapitel 4 dokumentiert (siehe Seite 35 ff.). Bei jedem genannten Konzept sind Querverweise auf die jeweils relevanten Abschnitte der Analyse angegeben. Jene Konzepte, die auf Grund der Vorauswahl mit Hilfe der K.O.-Kriterien ausgeschieden sind, werden im Abschnitt 3.7 nochmals zusammen fassend aufgezählt.

3.1 Gesamtübersicht

Die Menge der Materialien zu Projekten in Wissenschaft und Praxis, die sich mit gleichen oder ähnlichen Fragen und Problemstellungen wie die Diplomarbeit befassen, ist schier unüberschaubar. Es ist schwierig, aus der großen Menge der verfügbaren Projekte die brauchbaren Konzepte heraus zu finden.

Der vorliegende Abschnitt dient dem Leser als Leitfaden durch die Phasen der Auswahl und Analyse von Konzepten in Literatur und Praxis. Es wird erläutert, nach welchen Schritten die Konzeptanalyse durchgeführt wurde (Abschnitt 3.1.1) sowie nach welchen Richtlinien die Konzepte ausgewählt und untersucht wurden (Abschnitt 3.1.2).

3.1.1 Vorgehensweise bei der Konzeptanalyse

Für die Erstellung der Konzeptanalyse in der Diplomarbeit schlug ich folgenden Weg ein, um die Konzepte und Literatur zuerst zu identifizieren und die relevanten Konzepte auszuwählen:

1. Verschaffen eines **Überblicks** zu Konzepten, Projekten, usw., die für das Themengebiet der Diplomarbeit relevant sind. (Vgl. den vorliegenden Abschnitt 3.1.)

2. Bildung von Kategorien für Konzepte, die im Schritt 1 gefunden wurden; Erarbeitung des *state of the art* bezüglich der Einteilung des Themengebiets. (Vgl. Abschnitt 3.2.)
3. Einordnung aller gefundenen Konzepte in die zuvor gebildeten **Kategorien**. (Vgl. Abschnitte 3.3 bis 3.6.)
4. **Vorauswahl** von brauchbaren Konzepten auf Grund von K.O.-Kriterien. (Vgl. Abschnitte 3.3 bis 3.6 sowie Abschnitt 3.7.)
5. **Analyse** aller als brauchbar beurteilten Konzepte. (Vgl. Kapitel 4.)
6. **Bewertung** der verbliebenen Konzepte (nach Kategorien) auf ihre Tauglichkeit für eine Architektur des Contentpools. (Vgl. Kapitel 5.)
7. Entwerfen der Architektur des Contentpools unter Berücksichtigung der nach der Bewertung als am brauchbarsten beurteilten Konzepte. (Vgl. Kapitel 6.)

3.1.2 Richtlinien für die Auswahl von Konzepten

Bei der Auswahl von Konzepten wurden **zwei Richtlinien** beachtet, um eine möglichst vollständige Bestandsaufnahme des zu den Fragestellungen der Diplomarbeit vorhandenen Wissens zu gewährleisten. Die beiden Richtlinien lauten wie folgt:

- Der aktuelle **Stand der Forschung** („*state of the art*") auf allen für die Beantwortung der Fragestellungen relevanten Fachgebieten der Wissenschaft sollte berücksichtigt werden. Besonders im Mittelpunkt des Interesses stand der Wissensstand in Bezug auf Datenbanken, semantischen Netze und Java-Technologien für die Darstellung von komplexen Graphen.
- Auf Grundlage einer **Analyse der E-Learning-Märkte** (d.h. Märkte für Wissenstransfer-Umgebungen und für Lernmaterialien in Wissenstransfer-Umgebungen) wurden die kommerziell erfolgreichsten Produkte identifiziert. Aus den kommerziell erfolgreichen Produkten sollten die Konzepte, soweit möglich, identifiziert und analysiert werden.

Um dem Leser einen Gesamtüberblick zu geben, werden in der Folge alle Konzepte aufgezählt, die unter Beachtung der beiden genannten Richtlinien identifiziert wurden.

Die **Reihenfolge bei der Aufzählung** der Konzepte orientiert sich an der alphabetischen Sortierung, da zum Zeitpunkt der Konzeptauswahl noch keine Reihung nach jedweder Skala vorgenommen werden soll. In der Literatur wird den Konzepten häufig eine relative Bedeutung zugeordnet. Diese Zuordnungen spiegeln nicht selten die subjektive Meinung des jeweiligen Autors wider. Bei Konzepten, die aus kommerziellen Produkten identifiziert wurden, ist eine Reihenfolge ebenfalls schwer herstellbar.

Hinweis: vom Autor werden alle Konzepte im Abschnitt 3.2 (Seite 29 ff.) in Kategorien eingeteilt, d.h. nach Themengebieten sortiert, um das vorhandene Material zu ordnen und den Überblick zu erleichtern.

An Hand von Literatur, die zu Konzepten auf dem Gebiet des E-Learning einen Überblick gibt (vgl. dazu [RobCol, 2002], [Paulsen, 2003], [Svensson, 2001]), wurde der aktuelle

Stand der Forschung in relevanten Fachgebieten ermittelt. Aus der Literatur wurden folgende Konzepte für eine genauere Berücksichtigung in Betracht gezogen:[4]

- AICC Specifications (AICC = Aviation Industry CBT Committee)
- Educational Modelling Language (EML)
- Gateway to Educational Materials (GEM)
- IEEE Learning Object Model
- IEEE Learning Object Metadata Standard (LOM)
- IMS Metadata Specification
- IMS Content Packaging Specification
- Learning Material Markup Language (LMML)
- Schools Interoperability Framework (SIF)
- Sharable Content Object Reference Model (SCORM)
- XML Topic Maps (XTM)

Aufgrund der Analyse der E-Learning-Märkte wurden folgende Konzepte identifiziert:

- COMpendis
- Microsoft Learning Resource Interchange (LRN)
- Slicing Book Technology (SBT)
- Trial Solution

3.2 Kategorisierung der Konzepte

R. Robson und G. Collier geben ein **funktionales Modell** von Wissenstransfer-Umgebungen an, in dem alle wichtigen Komponenten einer dem Stand der Technik entsprechenden Wissenstransfer-Umgebung enthalten sind (siehe Abbildung 3). Dieses funktionale Modell kann als die **ideale Architektur** einer Wissenstransfer-Umgebung betrachtet werden (vgl. [RobCol, 2002] und [Collier, 2002]).

Wie später im Kapitel 6 sowie bei [Fenneberg, 2002], [Fürlinger, 2003] und [Radmayr, 2003] dargestellt wird, orientiert sich auch die Architektur der Wissenstransfer-Umgebung Scholion WB+ an dieser idealen Architektur.

[4] Hinweis: Referenzen zu allen genannten Konzepten folgen in den Abschnitten 3.3 bis 3.7. Die hier gebrachte Aufzählung hilft dem Leser, sich einen Überblick zu verschaffen.

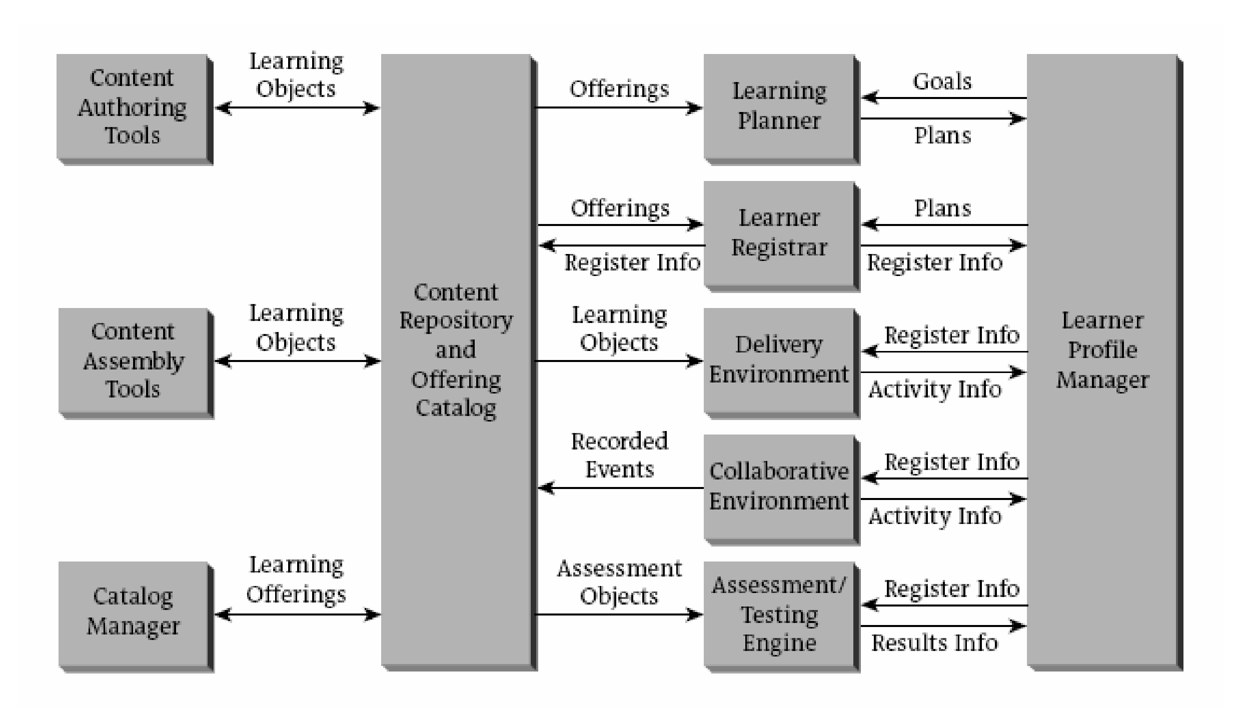


Abbildung 3: Funktionales Modell von Wissenstransfer-Umgebungen (Quelle: nach Eduworks Corporation[5]; zitiert in [RobCol, 2002])

Wie in der Abbildung 3 dargestellt, besteht eine Wissenstransfer-Umgebung im Idealfall aus den folgenden Komponenten (nach [Collier, 2002]):

- Wissensbasis mit einer Form von Inhaltsverzeichnis (*Content Repository*, *Offering Catalog*)
- Werkzeuge für die Erstellung und Katalogisierung von Inhalten der Wissensbasis (*Content Authoring Tools*, *Content Assembly Tools*, *Catalog Manager*)
- Komponenten zur Erstellung eines Lernenden-Modells (*Learner Profile Manager*)
- Verschiedene Werkzeuge für die Nutzung der Inhalte in der Wissensbasis der Wissenstransfer-Anwendung (*Learning Planner*, *Learner Registrar*, …)

Die in der Abbildung 3 im linken Teil dargestellten Komponenten „*Content Repository*“ und „*Content Authoring Tools*, *Content Assembly Tools*, *Catalog Manager*“ entsprechen der **Funktionalität**, die durch den **Contentpool** abzudecken ist. Der Rest der in der Abbildung gezeigten Komponenten ist für den Contentpool im Sinne der Diplomarbeit nicht relevant. Aus den in Abbildung 3 gezeigten **Schnittstellen** lassen sich Ansatzpunkte ableiten, für die eine Definition von Konzepten oder eine Standardisierung in Frage kommt.

Für eine Kategorisierung des wissenschaftlichen Materials zu den für den Contentpool wichtigen Problemstellungen sind folglich alle in Abbildung 3 in der linken Hälfte dargestellten Komponenten und Schnittstellen relevant, nämlich:

[5] Siehe http://www.eduworks.com

- Konzepte für das Design von **Wissensbasen**;
- Konzepte für das Design von **Werkzeugen** zur Verwaltung einer Wissensbasis;
- Konzepte und Standards für die **Daten**, die zwischen den Werkzeugen und einer Wissensbasis ausgetauscht werden müssen.

Diese Aufzählung dient als Grundlage für die Bildung von Kategorien des wissenschaftlichen Materials, das für den Contentpool interessant ist. Zusätzlich werden die im Kapitel 2 genannten Frage- und Problemstellungen als Richtlinien für die Kategorisierung verwendet. Als Ergebnis dieser Überlegungen wurden folgende **Kategorien** für Konzepte, Standards und Technologien gebildet:

- Konzepte zur Zerlegung von Dokumenten (Abschnitt 3.3)
- Standards und Konzepte zur Speicherung von Wissensatomen (Abschnitt 3.4)
- Standards und Spezifikationen für den Aufbau semantischer Netze (Abschnitt 3.5)
- Technologien für webbasierte Applikationen (Abschnitt 3.6)

In der Folge werden alle im vorigen Abschnitt 3.1 aufgezählten Konzepte in die genannten Kategorien **eingeteilt**. Weiters werden jene Konzepte ausgewählt, für die eine weitere Analyse durchzuführen ist. Die Ausführungen zu einer Kategorie befinden sich in jenem Abschnitt, der jeweils in Klammern genannt ist.

3.3 Konzepte zur Zerlegung von Dokumenten

Das Gebiet der automatischen, d.h. auf Basis eines Algorithmus von einer Maschine durchgeführten, Zerlegung von Dokumenten ist bis jetzt großteils Neuland für die theoretische Forschung. Lediglich ein Konzept konnte gefunden werden. Dieses ist mehr ein praktisches denn ein theoretisches Konzept:

- Slicing Book Technology (vgl. Abschnitt 4.2.1, Seite 43 ff.)

3.4 Standards und Konzepte zur Speicherung von Wissensatomen

Zur Speicherung von „Dokument-Trümmern" ist die theoretische Forschung bereits weiter fortgeschritten als zur Zerlegung an sich. Neben einem bereits existierenden, international anerkannten Standard konnten noch mehrere weitere Konzepte mit meist theoretischem, teilweise aber auch praktischem Bezug gefunden werden:

- IEEE Learning Objects (vgl. Abschnitt 4.3.1, Seite 53 ff.)
- Metadaten zu IEEE Learning Objects (vgl. Abschnitt 4.3.2, Seite 54 ff.)
- IMS Spezifikationen (vgl. Abschnitt 4.3.3, Seite 57 ff.)
- COMpendis Forschungsprojekt (vgl. Abschnitt 4.3.4, Seite 67 ff.)

3.5 Standards und Spezifikationen für den Aufbau semantischer Netze

Semantische Netze stellen ein theoretisch zwar erforschtes Gebiet dar, sind in der Praxis aufgrund der komplexen Anforderungen an deren Implementierung noch kaum vorhanden, geschweige denn erprobt. Nichts desto Trotz konnten Konzepte und Projekte identifiziert werden, die sich mit der Erstellung und Speicherung von semantischen Netzen befassen:

- Educational Modeling Language (EML – vgl. Abschnitt 4.4.1, Seite 70 ff.)
- XML Topic Maps (vgl. Abschnitt 4.4.2, Seite 77 ff.)
- Trial Solution Forschungsprojekt (vgl. Abschnitt 4.5.1, Seite 93 ff.)
- Sharable Content Objects Reference Model (SCORM – vgl. Abschnitt 4.5.2, Seite 100 ff.)

3.6 Technologien für webbasierte Applikationen

Wie bereits im Abschnitt 1.1 (Ziele dieser Diplomarbeit, Seite 2) erwähnt, stellt die Konzentration auf die Programmiersprache Java eine Vorbedingung des Projekts Scholion WB+ dar. Aus diesem Grund konnte die Auswahl der Technologien für webbasierte Anwendungen auf wenige Punkte beschränkt werden:

- Java Servlets
- Java Server Pages (JSP)

Auf eine Analyse der zuletzt genannten Technologien wird in der Diplomarbeit aus Platzgründen verzichtet. Der interessierte Leser wird auf [Sun, 1999] verwiesen.

3.7 Nicht analysierte Konzepte

Die Zahl der in Wissenschaft und Praxis vorhandenen Konzepte, die in Zusammenhang mit den Problemstellungen dieser Diplomarbeit stehen, ist groß. Die **Vorauswahl** und Einteilung in zu analysierende und nicht zu analysierende Konzepte war deshalb unerlässlich, um den Umfang der Analyse in einem überschaubaren Rahmen zu halten.

Beim Prozess der Identifikation von geeigneten Konzepten wurde daher vom Autor eine Vorselektion durchgeführt und ungeeignet scheinende Konzepte noch vor Beginn der Literatur- und Konzeptanalyse verworfen. Die Vorselektion der Konzepte wurde aufgrund von zuvor definierten, so genannten **K.O.-Kriterien** vorgenommen. Im Kapitel 5: Bewertung der Konzepte und Technologien, Seite 123 ff. findet der Leser die K.O.-Kriterien ausführlich beschrieben (Abschnitt 5.3.1, Seite 130 ff.).

Während dieser ersten Bewertung an Hand von K.O.-Kriterien wurden folgende beiden Schritte vollzogen:

- **Einordnung** der Konzepte in die zuvor (aus der Einteilung der Frage- und Problemstellungen) gebildeten Kategorien „Zerlegung von Dokumenten“, „Speicherung

von zerlegten Dokumenten", „Darstellung zerlegter Dokumente" und „Referenzprojekte und Rahmenmodelle".

- **Ausscheidung** von Konzepten, die auf Grund der Bewertung an Hand der K.O.-Kriterien als wenig brauchbar für eine weitere Analyse eingestuft wurden.

Folgende Konzepte wurden gemäß der K.O.-Kriterien als ungeeignet zur Erfüllung der Ziele der vorliegenden Arbeit identifiziert und nicht in die Analyse mit einbezogen:

- LMML (Learning Material Markup Language).[6]
- ARIADNE: ein auf IEEE LOM (vgl. Abschnitt 4.3.2, Seite 54 ff.) basierendes Projekt, das sich mit der Forschung an e-Learning Technologien befasst.[7]
- SIF (Schools Interoperability Framework).[8]
- Spezifikationen, Richtlinien und Empfehlungen von AICC (Aviation Industry CBT Committee).[9]
- GEM (Gateway to Educational Materials): eine Sammlung unterschiedlicher Metadaten-Standards.[10]
- Microsoft Learning Resource Interchange (LRN)[11]
- PROMETEUS (PROmoting Multimedia access to Education and Training in EUropean Society): ein Forum zur Diskussion von wissenschaftlichen oder praktischen Problemstellungen in verschiedenen Interessensgruppen.[12]

3.8 Zusammenfassung

Im Kapitel 3 wurden sowohl aus der Fachliteratur als auch aus Produkten der E-Learning-Märkte die gebräuchlichsten und erfolgreichsten Konzepte und Technologien identifiziert, die für die Diplomarbeit relevant sind. Mit diesem Vorgehen wurde das Ziel, den *state of the art* bei der Entwicklung des Contentpool-Konzepts zu berücksichtigen, verfolgt. Die Auswahl der Konzepte im Kapitel 3 definiert den Stand der Erkenntnis, der für die von der Diplomarbeit hinterfragten Fragestellungen gültig ist.

Folgende Kategorien von Konzepten wurden gebildet, um die Beantwortung der entsprechenden Fragestellungen der Diplomarbeit zu unterstützen:

- Konzepte zur Zerlegung von Dokumenten, die die Optimierung eines Werkzeugs für die Erstellung von Wissenseinheiten ermöglichen;

---

[6] Siehe http://www.lmml.de

[7] Siehe http://www.ariadne-eu.org/

[8] Siehe http://www.sifinfo.org/index.asp

[9] Siehe http://www.aicc.org

[10] Siehe http://www.geminfo.org

[11] Siehe http://www.microsoft.com/elearn/support.asp

[12] Siehe http://www.prometeus.org

- Konzepte zur Speicherung von Wissenseinheiten, sowie …
- Konzepte für den Aufbau und die Speicherung semantischer Netze, welche die Optimierung des Datenmodells für den Contentpool unterstützen;
- Konzepte und Technologien für web-basierte Java-Applikationen, um eine optimale Benutzungsschnittstelle für die Darstellung und Verwendung von Wissensatomen zu gewährleisten.

4 Related Work

Im Kapitel 3 wurden jene Fachgebiete der Wissenschaft abgesteckt, deren Stand der Erkenntnis für die Beantwortung der Fragestellungen der Diplomarbeit berücksichtigt werden muss. Darüber hinaus wurden relevante Produkte und Projekte aus den E-Learning-Märkten berücksichtigt. Das Kapitel 4 beschäftigt sich damit, den **Stand des Wissens** aus den Konzepten zusammen zu fassen. Damit wird das Ziel der Diplomarbeit, den *state of the art* bei der Entwicklung eines Contentpool-Konzepts zu berücksichtigen, erfüllt.

Die Ergebnisse der Literatur- und Konzeptanalyse werden vorgestellt. Es konnten zahlreiche viel versprechende Konzepte gefunden werden. In der Diplomarbeit werden nur jene Konzepte, Ansätze, Forschungsprojekte usw. untersucht, deren Relevanz für das Projekt „Entwicklung eines Contentpools für Scholion WB+" ausreichend gegeben war. D.h. solche Konzepte, die aus Sicht des Autors dabei helfen, die Ziele der Diplomarbeit (vgl. Abschnitt 1.1) zu erreichen. Meist sind dies Konzepte, in denen weniger die Entwicklung einer Wissenstransfer-Umgebung als das **Management von Content** (im weitesten Sinn) im Mittelpunkt steht.

Es sei explizit erwähnt, dass über die hier analysierten Konzepte hinaus noch zahlreiche weitere Forschungsansätze und Projekte existieren. Hierzu wird auf den Abschnitt 3.7 (Nicht analysierte Konzepte) im vorangegangenen Kapitel (siehe Seite 32) sowie auf das Literaturverzeichnis (Kapitel 10.1 am Ende dieses Dokuments, ab Seite 322) verwiesen. Vor allem [RobCol, 2002] gibt einen guten Überblick über alle bedeutenden Organisationen sowie über Konzepte, Spezifikationen und Standards.

Die Literaturanalyse wird in folgende Abschnitte eingeteilt:

4.1 Einführung in die Literaturanalyse

Im Projekt „Entwicklung eines Contentpools für Scholion WB+" war es nicht Ziel, „das Rad neu zu erfinden". Durch die Literaturanalyse wurde ein Überblick erarbeitet, inwieweit die Problemstellungen des Projekts durch **bereits realisierte Konzepte** schon gelöst wurden. Brauchbare Konzepte aus der Literatur sollten übernommen werden.

Eigene Lösungen sind nur für solche Problemstellungen erforderlich, für die im Rahmen von Projekten in Wissenschaft und Praxis noch keine Lösungen oder Lösungsansätze entwickelt wurden. Die Literaturanalyse dient also dazu, den Rahmen für den **Bedarf an eigenen Lösungen** abzustecken und die **Grundlage für eine Architektur** des Contentpools aus bewährten Konzepten zu bilden.

Auf den folgenden Seiten wird als Einführung in die Literaturanalyse, bevor die Ergebnisse dargelegt werden, die allgemeine Terminologie erläutert (Abschnitt 4.1.1). Danach wird eine Gliederung der Analyse vorgenommen (Abschnitt 4.1.2). Schließlich gehe ich auf die Ziele der Analyse ein (Abschnitt 4.1.3).

4.1.1 Allgemeine Terminologie

Im vorliegenden vierten Kapitel werden Konzepte präsentiert, die ähnliche oder gleiche Ziele verfolgen wie das in der Diplomarbeit beschriebene Projekt „Entwicklung eines Contentpools für Scholion WB+". Um die begriffliche Konsistenz zu wahren, wurde ein **einheitliches System von Begriffen** („Terminologie") verwendet. Dies erleichtert die Beschreibung der Konzepte, vor allem den Vergleich verschiedener Konzepte gegeneinander. Der folgende Abschnitt stellt die Terminologie vor.

Selbstverständlich orientiert sich die Terminologie, die zur Analyse von verwandten Konzepten gebraucht wurde, am **Glossar**, das am Ende dieses Dokuments zu finden ist (Abschnitt 10.2, Seite 333 ff.). An dieser Stelle sei daher nochmals betont, dass wichtige Begriffe aus der Terminologie der Diplomarbeit im Glossar definiert werden. Auf eine nochmalige Aufzählung der Begriffe im vorliegenden Kapitel verzichte ich aus Platzgründen.

Häufig musste die in den Originalquellen verwendete Begriffswelt erst in die Terminologie der Diplomarbeit übertragen werden. Dies geschah, wie oben erläutert, um den Vergleich zwischen Konzepten zu erleichtern und eine **objektive Beschreibung und Analyse** zu ermöglichen. Daher kann und wird es vorkommen, dass in zahlreichen von mir verwendeten Quellen von den Autoren andere Begriffe verwendet wurden.

Exkurs: Unterscheidung zwischen „Spezifikation" und „Standard"

Von großer Bedeutung für die Literaturanalyse ist die Klärung des Unterschieds zwischen den häufig verwendeten Begriffen „Konzept", „Ansatz", „Standard" und „Spezifikation". Da der Unterschied nicht einfach zu zeigen ist, wurde diese Unterscheidung aus dem Glossar heraus genommen und wird in der Folge behandelt. Besonders für die Definition der Ziele für die Literaturanalyse (vgl. Abschnitt 4.1.2, Seite 40) muss die unterschiedliche Semantik dieser Begriffe definiert und erläutert werden.

Im Glossar finden sich die Definitionen zu den einzelnen Begriffen „Konzept", „Ansatz", „Spezifikation" und „Standard" (siehe Kapitel 10.2). Zur Definition des Begriffs „Konzept" siehe Seite 339; die Begriffe „Spezifikation" und „Standard" sind auf der Seite 341 definiert. Der Begriff „Ansatz" wird synonym zu „Konzept" gebraucht, ist aber dennoch auf Seite 334 explizit definiert.

Mit „Konzept" werden alle in Wissenschaft und Praxis gebräuchlichen **Projekte** bezeichnet, deren Inhalte für das Projekt „Entwicklung eines Contentpools für Scholion WB+" und die Diplomarbeit von Interesse waren. „Konzept" und „Ansatz" werden folglich als **Oberbegriff** für Spezifikationen und Standards genommen, die aus Projekten hervorgehen. Schwieriger ist zu definieren, was mit „Spezifikation" bzw. „Standard" gemeint ist.

Am präzisesten lassen sich die **Unterschiede** zwischen Spezifikation und Standard an Hand von Merkmalen wie „Zeitdauer für die Entwicklung" oder „Bedeutung für Forschung

und Praxis" definieren. Die typischen Merkmalsausprägungen für beide Varianten eines Ansatzes sind in Tabelle 1 aufgelistet: [Quelle: nach Oracle Corporation, IMS Basic Training: Content Packaging Module 2. http://ilearning.oracle.com]

Tabelle 1: Definition der Unterschiede zwischen „Spezifikation" und „Standard"

Ausprägungen / Merkmale	Spezifikation	Standard
Zeitdauer für Entwicklung	Schnelle Entwicklung: kurze Zeitdauer	Langsame Entwicklung: lange Zeitdauer
Rolle bei der Fertigung eines Produkts	„Ermöglicher"	„Regulierer"
Vorgehensweise bei der Entwicklung	Nimmt schnell grobe Übereinstimmungen auf	Zusammenlegung akzeptierter, erprobter Konzepte und Lösungen aus der Praxis
Status bezüglich möglicher Veränderungen	Werden experimentell weiterentwickelt	Endgültig

Abbildung 4 zeigt das **Vorgehensmodell** zur **Entwicklung von Standards**, das mit der Veröffentlichung von Spezifikationen beginnt. Der grundsätzliche Unterschied zwischen den beiden Begriffen „Standard" und „Spezifikation" wird aus dieser Abbildung (zusätzlich zu den Definitionen im Glossar) ersichtlich.

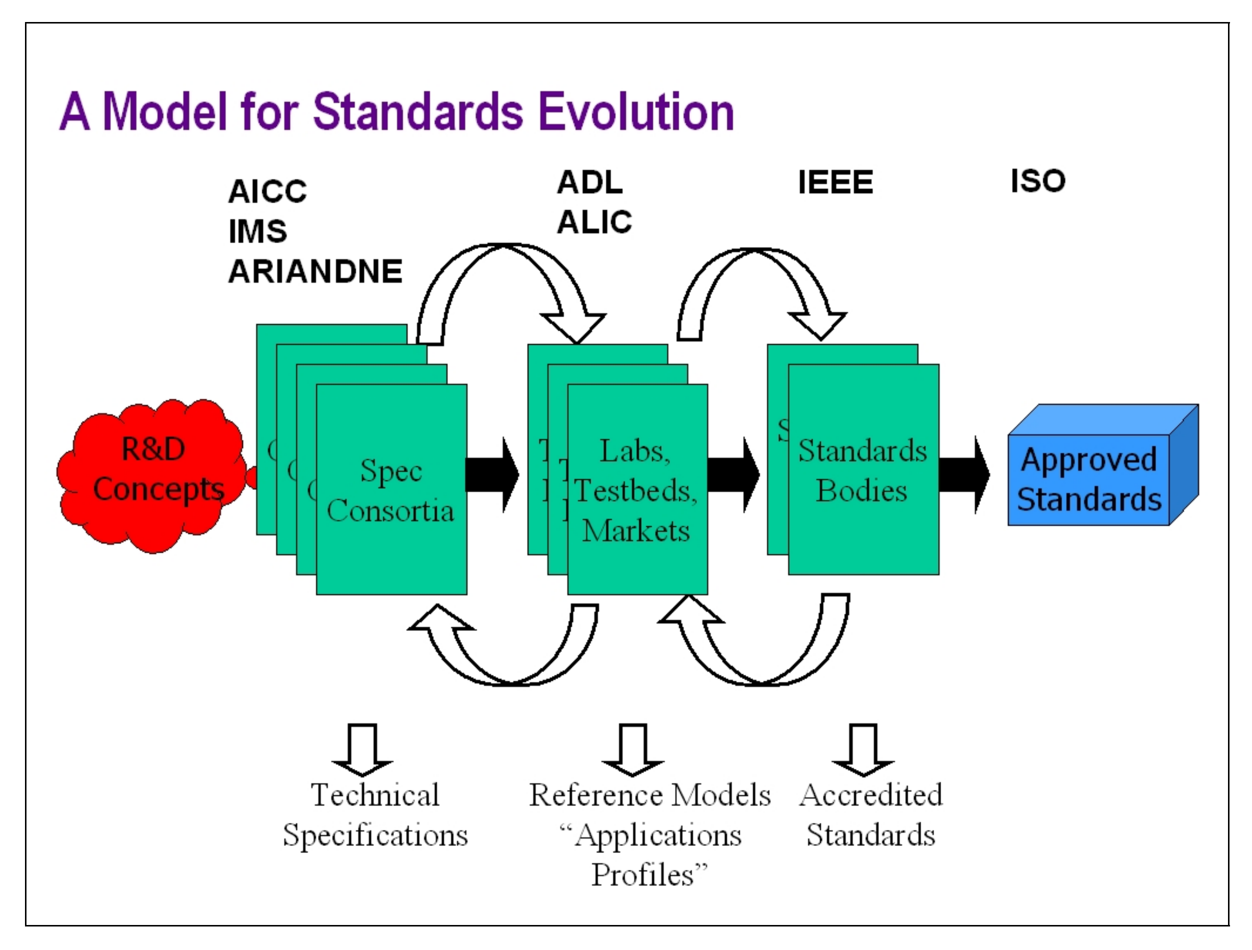


Abbildung 4: Vorgehensmodell für die Entwicklung von Standards [DoddsWest, 2002]

Wenn die Forderung nach **Offenheit** des zu implementierenden Systems erfüllt werden soll, führt kein Weg an der Berücksichtigung von anerkannten und verbreiteten Standards und Spezifikationen vorbei. Seit mehreren Jahren bemühen sich zahlreiche Organisationen, Konsortien und Komitees um die Schaffung von weltweit anerkannten Standards auf dem Gebiet des virtuellen Lernens (e-Learning).

Mehrere Forschungsprojekte befass(t)en sich zudem mit der Entwicklung von **Tools** zur Erstellung und Verwaltung von Content. Ein Ende der Forschungen ist bei weitem nicht vorhersehbar. Noch konnte sich jedoch kein Vorschlag als Standard wirklich durchsetzen, da vor allem **empirische Daten** bezüglich der Akzeptanz durch die Benutzer oder der Effizienz des Datenaustausches fehlen.

Unumstritten sind die **positiven Auswirkungen**, falls die Schaffung von einheitlichen Standards gelingt. Dies beginnt damit, dass seit den Anfängen der Industrialisierung jeder Industriezweig, dem die Einigung auf eine Standardisierung gelang, danach in eine Boom-Phase eintrat. Mit anderen Worten markiert der Zeitpunkt, an dem die Einigung auf Standards vollzogen werden kann, den **Übergang von der Pionier-Phase**, in der nur wenige Organisationen den Umgang mit der neuen Technologie wagen, in eine Phase, die durch das Auftreten **vieler neuer Entwickler** und das **allmähliche Entstehen eines breiten Marktes** gekennzeichnet ist. Die Aktivitäten im Industriezweig verlagern

sich zunehmend von prototypischen Experimenten zur Entwicklung konkreter Anwendungen. Abbildung 5 stellt diesen Phasenübergang schematisch dar.

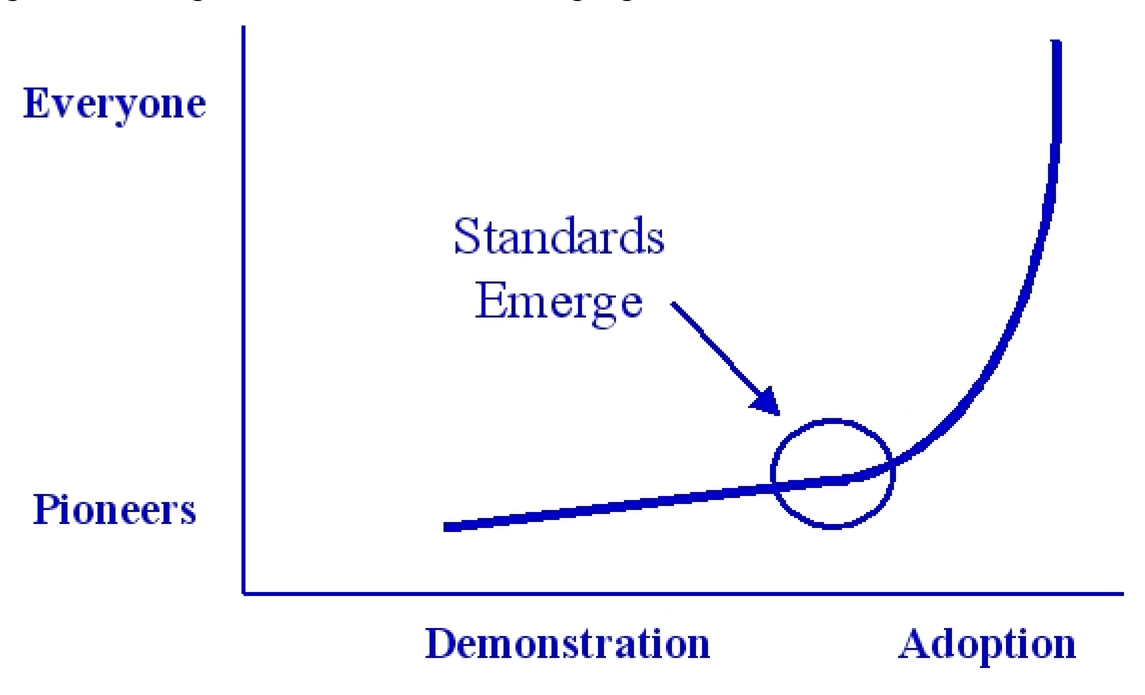


Abbildung 5: Industriestandards als Auslöser des Übergangs zur Boom-Phase [Robson, 2001]

Robson und Collier nennen zudem folgende Vorteile für den Bereich Wissenstransfer (e-Learning), wenn durch die Entwicklung anerkannter Standards eine hohe Interoperabilität erreicht werden kann [RobCol, 2002]:

- **Konsumenten**, d.h. Käufer einer Wissenstransfer-Umgebung, sind nicht von einem bestimmten Anbieter abhängig. Durch strenge **Modularisierung und Offenheit** der Wissenstransfer-Systeme ist es sogar denkbar, Komponenten verschiedener Hersteller miteinander zu kombinieren.
- Es lässt sich absehen, dass offene Standards und Systeme die Hemmungen der Konsumenten vor der Investition in Wissenstransfer-Anwendungen reduzieren. Daraus ergibt sich eine **höhere Nachfrage** nach Kursmaterialien und Nachschlagewerken (Content), so dass ein größerer und lukrativerer Markt für den **Handel mit Content** entsteht. Mit dem Anreiz steigt die Wahrscheinlichkeit, dass zahlreiche Hersteller wieder verwendbare Contents erzeugen und anbieten.
- Für die **Anbieter bzw. Entwickler** der Wissenstransfer-Umgebungen ergibt sich der Vorteil, dass durch Standards für den Austausch von Daten zwischen Systemkomponenten die **Entwicklungskosten reduziert** werden, da keine proprietären Schnittstellen entwickelt werden müssen.
- **Hersteller von Content** können sich darauf verlassen, dass ihre Kursmaterialien und Nachschlagewerken in **mehreren Wissenstransfer-Umgebungen** verwendbar sind.

- **Lernende**, d.h. die Konsumenten von Contents, werden zwischen zahlreicheren Produkten auswählen können. Zudem ist Information über Lernende zwischen verschiedenen Wissenstransfer-Umgebungen austauschbar.

4.1.2 Gliederung der Literaturanalyse

Um die Literaturanalyse systematisch und interindividuell nachvollziehbar durchzuführen, wird hier eine **Aufteilung** des zu untersuchenden wissenschaftlichen Materials **in Themenbereiche** vorgenommen. Aus den Ausführungen im Kapitel 3 (Auswahl der Konzepte und Technologien, Seite 27 ff.) wird ersichtlich, dass folgende Projekte und Konzepte für eine Literaturanalyse ausgewählt wurden:

- Slicing Book Technology (SBT)
- IEEE Learning Object Model
- IEEE Learning Object Metadata Standard
- IMS Global Learning Consortium Spezifikationen
- Educational Modeling Language (EML)
- COMpendis Forschungsprojekt
- XML Topic Maps (XTM)
- Trial Solution Projekt
- Sharable Content Object Reference Model (SCORM)

Die Reihenfolge der obigen Aufzählung darf nicht als Wertordnung jedweder Art gesehen werden, sondern entspricht der Reihenfolge ihrer Erwähnung im Kapitel 3.

Aus den im Abschnitt 2.2 (Seite 17 ff.) definierten Fragen und Problemstellungen dieser Diplomarbeit ergibt sich eine Gliederung der Literaturanalyse **nach Themenbereichen**. Folgende Themenbereiche lassen sich durch Gliederung logisch zusammen gehörender Teilfragen bilden:

- Konzeptanalyse – Zerlegung von Dokumenten (Abschnitt 4.1.2.1)
- Konzeptanalyse – Speicherung zerlegter Dokumente (Abschnitt 4.1.2.2)
- Konzeptanalyse – Darstellung und Verwendung zerlegter Dokumente (Abschnitt 4.1.2.3)
- Konzeptanalyse – Referenzprojekte und Rahmenmodelle (Abschnitt 4.1.2.4)

Hinweis: Die letzte der oben genannten Kategorien wurde für jene Projekte eingeführt, die sich nicht exakt in eine der drei erstgenannten Themenbereiche einordnen lassen. Diese Projekte stellen mehrere Konzepte vor, die sich in mindestens zwei der Themenbereiche einordnen lassen.

Gegenstand der Literaturanalyse ist die Erhebung des aktuellen Stands von Forschung und Praxis zu den oben aufgezählten Themenbereichen. Auf den folgenden Seiten werden die oben aufgezählten Projekte und Konzepte einem der vier Themenbereiche zugeordnet.

4.1.2.1 Konzeptanalyse – Zerlegung von Dokumenten

Die Konzeptanalyse zur Zerlegung von Dokumenten als Teil der Literaturanalyse hat folgende Fragestellungen zu beantworten (vgl. Abschnitt 2.2.1):

- Wie können die Stellen bestimmt werden, an denen der Inhalt eines Dokuments zerlegt werden soll?
- Welche Metadaten zu den Teilen eines zerlegten Dokuments müssen bestimmt (d.h. aus dem Dokument extrahiert) werden?

Aus der Einteilung der Konzepte (vgl. Kapitel 3) wird folgendes Konzept als zugehörig zum Themenbereich „Zerlegung von Dokumenten" (Abschnitt 4.2) erachtet:

- Slicing Book Technology (SBT) (vgl. Abschnitt 4.2.1, Seite 43 ff.)

4.1.2.2 Konzeptanalyse – Speicherung zerlegter Dokumente

Die Konzeptanalyse zur Speicherung zerlegter Dokumenten als Teil der Literaturanalyse hat folgende Fragestellungen zu beantworten (vgl. Abschnitt 2.2.2):

- Welche Daten müssen zu den Teilen eines zerlegten Dokuments gespeichert werden, so dass, wie gefordert, semantische Aspekte der Inhalte abgebildet werden können?
- Wie können Teile der zerlegten Dokumente gespeichert werden?
- Wie können Metadaten zu den zerlegten Dokumenten gespeichert werden?

Aus der Einteilung der Konzepte (vgl. Kapitel 3) werden folgende Konzepte als zugehörig zum Themenbereich „Speicherung zerlegter Dokumente" (Abschnitt 4.3) erachtet:

- IEEE Learning Object Model (vgl. Abschnitt 4.3.1, Seite 53)
- IEEE Learning Object Metadata Standard (vgl. Abschnitt 4.3.2, Seite 54 ff.)
- IMS Global Learning Consortium Spezifikationen (vgl. Abschnitt 4.3.3, Seite 57 ff.)
- COMpendis Forschungsprojekt (vgl. Abschnitt 4.3.4, Seite 67)

4.1.2.3 Konzeptanalyse – Darstellung und Verwendung zerlegter Dokumente

Die Konzeptanalyse zur Darstellung und Dokumenten als Teil der Literaturanalyse hat folgende Fragestellungen zu beantworten (vgl. Abschnitt 2.2.3):

- Wie kann eine Navigation durch gespeicherte Inhalte von zerlegten Dokumenten erfolgen?
- Wie kann eine Suchfunktion in den gespeicherten Inhalten realisiert werden?
- Wie können Inhalte und Metadaten zerlegter elektronischer Dokumente dargestellt werden?

Aus der Einteilung der Konzepte (vgl. Kapitel 3) werden folgende Konzepte als zugehörig zum Themenbereich „Darstellung und Verwendung zerlegter Dokumente" (Abschnitt 4.4) erachtet:

- Educational Modeling Language (EML) (vgl. Abschnitt 4.4.1, Seite 70 ff.)
- XML Topic Maps (XTM) (vgl. Abschnitt 4.4.2, Seite 77 ff.)

4.1.2.4 Konzeptanalyse – Referenzprojekte und Rahmenmodelle

Die Konzeptanalyse von Referenzprojekten und Rahmenmodellen als Teil der Literaturanalyse hat Fragestellungen zu beantworten, die aus mehreren oder allen der drei zuvor genannten Themenbereiche stammen.

Aus der Einteilung der Konzepte (vgl. Kapitel 3) werden folgende Projekte als zugehörig zum Themenbereich „Referenzprojekte und Rahmenmodelle" (Abschnitt 4.5) erachtet:

- Trial Solution Projekt (vgl. Abschnitt 4.5.1, Seite 93 ff.)
- Sharable Content Object Reference Model (SCORM) (vgl. Abschnitt 4.5.2, Seite 100 ff.)

4.1.3 Ziele der Literaturanalyse

Oberstes Ziel der Literaturanalyse war es, **bereits realisierte Konzepte** in Wissenschaft und Praxis zu finden, die sich mit einer oder mehrerer der **Fragestellungen dieser Diplomarbeit** (vgl. Kapitel 2: Ausgangssituation und Problemstellung, Seite 15) auseinander setzen. Die als wertvoll eracheten Konzepte werden für den Entwurf der Contentpool-Verwaltung heran gezogen.

Als Leitfaden für die Auswahl der Inhalte der Literaturanalyse (d.h. die Auswahl der untersuchten Literatur) dient die **Einteilung in Themenbereiche** (Abschnitt 4.1.2). Zu folgenden Themenbereichen sollte der Stand der Wissenschaft („state of the art") ermittelt werden:

- Konzeptanalyse – Zerlegung von Dokumenten (4.1.2.1)
- Konzeptanalyse – Speicherung zerlegter Dokumente (4.1.2.2)
- Konzeptanalyse – Darstellung und Verwendung zerlegter Dokumente (4.1.2.3)
- Konzeptanalyse – Referenzprojekte und Rahmenmodelle (4.1.2.4)

Alle Themenbereiche waren unter Beachtung der im Abschnitt 2.3 (Seite 18 ff.) definierten **Rahmenziele** für die Gestaltung des Contentpools zu analysieren. Als Rahmenziele wurden definiert:

- Beachtung ergonomischer Grundsätze und Erkenntnisse (Benutzbarkeitskriterien, vgl. 2.3.1)
- Plattform-Unabhängigkeit (vgl. 2.3.5)
- Unabhängigkeit von zusätzlicher Software (vgl. 2.3.5)
- Kompatibilität bzw. Interoperabilität (vgl. 2.3.5)

- Offenheit in den Schnittstellen und der Architektur (vgl. 2.3.5)
- Skalierbarkeit und Leistungsfähigkeit (vgl. 2.3.5)

Hinweis: Die in den Abschnitten 2.3.2, 2.3.3 und 2.3.4 behandelten Designziele fließen in die Definition des Zielsystems für die Literaturanalyse ein.

Die Identifikation und Verwendung von **Standards und Spezifikationen** ist als Grundlage der Architektur für den Contentpool von besonderer Wichtigkeit. Wie zuvor gezeigt (vgl. Abschnitt 4.1.1), können damit die Forderungen nach Offenheit und Interoperabilität des zu implementierenden Systems mit großer Wahrscheinlichkeit erfüllt werden. Damit wird letztlich das in der Informatik stets angestrebte Ziel von **Kompatibilität** zwischen verschiedenen Systemem verfolgt (vgl. I. Wende in [RechPom, 1999], S. 1083 – 1091). Im Zusammenhang mit der Implementierung des Contentpools steht im Speziellen das Ziel im Vordergrund, das System so weit wie möglich unabhängig von einer bestimmten Plattform oder einer bestimmten Systemkonfiguration zu realisieren.

Als grundlegendes Ziel, das in der Analyse aller Themenbereiche beachtet wurde, war daher die **vorrangige Behandlung von Standards** gegenüber von Spezifikationen zu beachten.

4.2 Konzepte zur Zerlegung von Dokumenten

In diesem Abschnitt werden Konzepte aus Wissenschaft und Praxis untersucht, die sich mit der Zerlegung elektronischer Dokumente beschäftigen. Gemäß der Gliederung der Literaturanalyse (siehe Abschnitt 4.1.2) werden im Folgenden behandelt:

4.2.1 Slicing Book Technology (SBT)

Von der Forschungsgruppe „Künstliche Intelligenz“ der **Universität Koblenz-Landau** wurde die so genannte Slicing Book Technology entwickelt. Ziel des Verfahrens ist die Wertsteigerung von Texten, insbesondere von Lehrbüchern und Fachtexten, durch **Digitalisierung und Zelegung**. Die Technologie konzentriert sich vor allem auf eine im Sinne der Wiederverwendbarkeit hochwertige Zerlegung. Konzepte zur Speicherung oder Nutzung der geschaffenen Wissensbasis fehlen weitgehend [Dahn, 2001a].

Im Rahmen der Slicing Book Technology werden Konzepte zur **Zerlegung von elektronischen Dokumenten**, v.a. von Büchern, und deren **Aufbereitung mit Metadaten** erforscht. Die Konzepte wurden in mehreren Werkzeugen umgesetzt und werden kommerziell vertrieben. Versionen der Werkzeuge für Forschungszwecke waren zum Zeitpunkt der Erstellung dieser Diplomarbeit leider nicht verfügbar.

Folgende Konzepte sind Gegenstand der Slicing Book Technology:

- Zerlegung elektronischer Bücher
- Sammlung von Metadaten zu zerlegten Büchern

Eine grundlegende Eigenschaft der Konzepte der Slicing Book Technology ist die Bearbeitung aller Vorgänge in jeweils **zwei Schritten**: zuerst automatisch, anschließend manu-

ell. Der manuelle Bearbeitungsschritt dient stets zur Verfeinerung und Ergänzung der automatisch genierten Ergebnisse. Durch Kombination der Konzepte mit den Attributen „automatisch“ und „manuell“ in einer Matrix lassen sich insgesamt vier Konzepte erkennen, die in der Folge präziser vorgestellt werden:

- Automatische Zerlegung des Quelldokuments (4.2.1.1)
- Automatisches Generieren der Metadaten (4.2.1.2)
- Manuelle Nachbearbeitung der Zerlegung des Dokuments (4.2.1.3)
- Manuelle Bearbeitung und Ergänzung der Metadaten (4.2.1.4)

Die in der Folge präsentierten Fakten wurden aus [Dahn, 2000a], [Dahn, 2001a], [Dahn, 2001b], [Dahn et al., 2001], [Lange, 2001] und [Valerius, 2001] entnommen. Für weitere Details zur präsentierten Information wird auf die genannten Quellen verwiesen; besonders [Dahn, 2001b] wird empfohlen.

4.2.1.1 Automatische Zerlegung des Quelldokuments

Der automatische Zerlegungsvorgang nimmt dem Benutzer so weit wie möglich die **Generierung der Wissenseinheiten** („Slices“ [Dahn, 2001b]) ab. Für den Zerlegungsvorgang sind die folgenden Schritte erforderlich (vgl. [Dahn et al., 2001]):

- Konfigurieren der Granularität
- Automatisches Erkennen von Grenzen zwischen Wissenseinheiten
- Konfiguration von Layoutregeln
- Automatisches Generieren eines Dateien-Baumes

Die genannten Schritte sind als Funktionalität in Werkzeugen der Slicing Book Technology verfügbar und werden nachfolgend erläutert.

4.2.1.1.1 Konfigurieren der Granularität

Dem Benutzer wird ermöglicht, die „Feinheit“ der Wissenseinheiten zu konfigurieren. Die **Feinheit** wird im Wesentlichen durch die Länge der in eine Einheit aufgenommenen Rohdaten bestimmt. Ein Messen der Feinheit ist nur durch Angabe **abstrakter Messeinheiten** möglich, z.B. ein ganzer Satz oder ein Absatz [Dahn, 2001b].

Mögliche Ausprägungen der Granularität befinden sich zwischen den beiden Extremen „eine Zeile“ bzw. „ein Kapitel“. Zur Zeit ist jedoch **noch keine Skala definiert**, die die Ausprägungen zwischen den beiden Extremen festlegt. Die sinnvollsten Größen für eine Wissenseinheit können zudem durch ein Werkzeug nicht bestimmt werden, sondern es liegt im Ermessen des Benutzers, diese zu definieren (vgl. [Dahn et al., 2001]).

Es ist intuitiv leicht nachzuvollziehen, dass der Splitter-Algorithmus der SBT wesentlich auf die vom Autor des Quelldokuments gewählten **Formatierungen** bzw. Einteilungen in Absätze angewiesen ist. Mit anderen Worten: je besser das Dokument nach semantischen Gesichtspunkten strukturiert ist, desto brauchbarer werden die Ergebnisse der automatischen Zerlegung sein. Umgekehrt lässt sich behaupten, dass die Zerlegung auf

Basis eines Algorithmus umso schwieriger wird, je schlechter das Quelldokument strukturiert ist [Dahn, 2001b].

4.2.1.1.2 Automatisches Erkennen von Grenzen zwischen Wissenseinheiten

Im Algorithmus zur automatischen Zerlegung von Dokumenten in Wissenseinheiten müssen **Regeln fixiert** sein, die eine Erkennung von Grenzen zwischen potentiellen Wissenseinheiten erlauben. Die große Schwierigkeit hierbei besteht darin, einen **möglichst brauchbaren Output** zu erzeugen, d.h. die Grenzen zwischen den Wissenseinheiten so zu generieren, dass die erzeugten Wissenseinheiten semantisch in sich geschlossen sind.

Bis dato sind die Möglichkeiten der Texterkennung noch nicht so weit fortgeschritten, dass eine Zerlegung auf semantischer Ebene durchführbar ist. Die Slicing Book Technology behilft sich daher mit Erkennungsregeln auf **syntaktischer Ebene**. Folgende Einheiten eines Textes eignen sich als Grenzen zwischen Wissensatomen [Dahn, 2000a]:

- Kapitelüberschriften und Unterkapitelüberschriften
- Beschriftungen von Abbildungen, Tabellen, usw.
- Abbildungen
- Tabellen
- Codefragmente

Die obige Aufzählung erhebt nicht den Anspruch auf Vollständigkeit, sondern soll lediglich die wichtigsten Beispiele anführen. Zwei wichtige Schlussfolgerungen lassen sich aus den bisherigen Feststellungen ziehen:

- **Formatierungen** von Texten sind heutzutage leider bei Weitem nicht einheitlich. Es ist daher mit großen Schwierigkeiten verbunden, **konkrete Regeln zu formulieren**, an welchen Merkmalen sich die Bausteine eines Textes erkennen lassen.
- Je nachdem, aus welchem Fachgebiet das Quelldokument stammt, ändern sich auch die **denkbaren Einheiten**, die innerhalb eines Textes vorkommen können. Als letztes Beispiel wurde oben „Codefragmente" angeführt; solche Elemente werden mit großer Wahrscheinlichkeit nur in Dokumenten vorkommen, die aus dem Fachgebiet der Informatik stammen.

Die beiden eben genannten Probleme stellen die größte Herausforderung dar, die es bei der Definition von Regeln zur Erkennung von Grenzen zwischen Wissenseinheiten gibt. Ein möglicher Lösungsansatz für das erste Problem stellt die Definition von so genannten **Layoutregeln** dar. Diese werden im folgenden Punkt 4.2.1.1.3 behandelt.

Das zweite Problem, die fachspezifische Ausprägung von semantischen Texteinheiten, kann wohl nur durch die Erstellung eigener **Strukturdefinitionen** gelöst werden. Eine Strukturdefinition enthält Metainformation über Textbausteine eines bestimmten Fachgebietes. Es muss damit gerechnet werden, dass Strukturdefinitionen nicht ohne Weiteres auf andere Fachgebiete übertragbar sind. Ihre Definition muss folglich durch einen Experten auf dem jeweiligen Fachgebiet erfolgen (vgl. dazu [Lange, 2001]).

4.2.1.1.3 Konfiguration von Layoutregeln

Unter Layoutregeln wird die Definition von Formatierungen für ein bestimmtes Textelement verstanden, mit dessen Hilfe sich in Dokumenten eines bestimmten Verfassers das **Textelement identifizieren** lässt. Man muss dabei von der These ausgehen, dass ein Autor die von ihm verfassten Texte stets sehr ähnlich, wenn nicht sogar identisch, formatiert (vgl. [Dahn, 2001b]).

Es ist dabei nicht nur denkbar, dass einzelne Autoren stets eine bevorzugte Layoutierung verwenden, sondern es könnte auch sein, dass ein Verlag seinen Autoren die Formatierungsregeln vorgibt. Wie auch immer; mit Hilfe der Layoutregeln lässt sich die Trefferquote bei der Erkennung semantischer Texteinheiten erhöhen.

Folgende Features müssen gewährleistet werden, um den Algorithmus zur Zerlegung elektronischer Bücher bestmöglich zu unterstützen [Dahn, 2001b]:

- Definition von **Layoutregeln** für einen bestimmten **Autor**
- Definition von Layoutregeln für einen bestimmten **Verlag**
- Persistente **Speicherung** von definierten Layoutregeln zum Zweck der Wiederverwendung
- Definition von **Default-Einstellungen**; das sind jene Layoutregeln, bei denen die Wahrscheinlichkeit am größten ist, dass sie bei allen Dokumenten, die zur Anwendung des Zelegungsvorgangs in Frage kommen, zumindest brauchbare, wenn schon nicht optimale Ergebnisse liefern.

4.2.1.1.4 Automatisches Generieren eines Dateien-Baumes

Nach Abschluss der automatischen Erkennung von Grenzen zwischen Wissenseinheiten baut der Splitter einen Dateienbaum auf, der eine **Repräsentation der generierten Wissenseinheiten** im Arbeitsspeicher und im Sekundärspeicher ermöglicht. Ein Dateienbaum der SBT besteht aus den beiden Entitäten Blatt(knoten) und innerer Knoten (vgl. [Dahn, 2000a], [Dahn, 2001b]).

- Ein **Blatt** ist definiert als eine Datei, die genau eine Wissenseinheit enthält.
- Ein **innerer Knoten** ist eine Zusammenfassung mehrerer anderer Knoten; d.h. er enthält Referenzen auf Knoten, die sich in der Baumhierarchie unter ihm befinden.
- Jeder Knoten besitzt **drei Dateien**. Eine davon speichert den Inhalt. Die beiden anderen, das **Start-File** und das **End-File**, enthalten Strukturinformation. Start- und End-Files gibt es, um die technische Wohlgeformtheit des Dateien-Baumes sicherstellen zu können ([Dahn, 2001b], S. 13).

4.2.1.2 Automatisches Generieren der Metadaten

Metadaten im Kontext der Slicing Book Technology ist all jene Information, die zur Beschreibung einer Wissenseinheit dient. Die Aufbereitung der aus einem elektronischen Dokument erzeugten Wissenseinheiten mit Metadaten stellt sowohl den **aufwendigsten** als auch den für die Erhöhung der Wiederverwendbarkeit **wichtigsten Schritt** beim Auf-

bau einer Wissensbasis dar. Folgende Arten von Metainformation können automatisiert (mit Hilfe eines Algorithmus) erkannt werden (vgl. [Dahn, 2000a], [Dahn, 2001a], [Dahn, 2001b]):

- Information über die **Struktur** des Dokuments (Titel, Kapitel-Nr., Seite, ...)
- **Kategorisierung** – welche Art von semantischer Einheit liegt vor? (Z.B. Fließtext, Abbildung, Tabelle, ...)
- Daten zur Wahrung des **Urheberrechts** (Name des Autors, Name des Verlags, ursprüngliche Position innerhalb des Dokuments, ...)
- **Beziehungen** zwischen Wissenseinheiten
- **Technische** Information

Im Folgenden wird auf die genannten Kategorien von Metainformation noch präziser eingegangen. Es wird dabei erläutert, welche Konzepte zur Ermittlung dieser Daten im Rahmen der Slicing Book Technology zum Einsatz kommen.

4.2.1.2.1 Strukturinformation

Information über die Struktur eines Dokuments, d.h. seine Gliederung, sind am Einfachsten automatisiert erfassbar. Es werden folgende Daten gespeichert [Dahn, 2001d]:

- Titel des Dokuments
- Kapitel-Überschriften
- Kapitel-Nummerierungen
- Seiten-Nummer
- Eindeutige ID der Wissenseinheit

4.2.1.2.2 Kategorisierung

Die Kategorisierung gibt an, um welchen **Typ von semantischer Einheit** es sich bei der gegebenen Wissenseinheit handelt. Bereits im Abschnitt 4.2.1.1.2 („Automatisches Erkennen von Grenzen zwischen Wissenseinheiten") wurde darauf hingewiesen, dass für jedes Fachgebiet spezifische Arten von Wissenseinheiten definiert werden sollten. Folgende Daten zur Kategorisierung sind folglich zu speichern:

- Referenz auf die Definition gültiger Bezeichnungen für Kategorien
- Bezeichnung der Kategorie

Zu beachten ist nach [Dahn, 2001d], dass die Zuordnung einer Wissenseinheit nicht nur zu einer, sondern auch zu mehreren Kategorien möglich ist.

4.2.1.2.3 Urheberrechtliche Information

Es ist für die **praktische Verwendbarkeit** der Wissensbasis ungemein wichtig, dass urheberrechtliche Aspekte berücksichtigt werden können [Dahn, 2001d]. Ein Autor könnte z.B. verbieten, dass die Reihenfolge der Anordnung seines Textes verändert wird; er

könnte die gemeinsame Veröffentlichung seiner Publikationen mit denen anderer Autoren untersagen; usw.

Ein weiterer wichtiger Aspekt in diesem Zusammenhang stellt die Möglichkeit dar, Daten zur **Entgeltverrechnung gemäß des Urheberrechts** zur Verfügung zu haben. D.h. wenn ein Nutzer der Wissensbasis eine Wissenseinheit zur Erstellung eines Dokuments verwenden möchte, muss er wissen, an wen das Entgelt in welcher Höhe zu verrichten ist [Dahn, 2001d].

Folgende Information ist deshalb unbedingt zu speichern:

- Kosten der Wissenseinheit oder des Dokuments
- Restriktionen bezüglich des Gebrauchs und der Weiterverwendung
- Rechteinhaber, d.h. Autor und/oder Verlag
- Ursprüngliche Position der Wissenseinheit im Originaldokument

Anmerkung: für eine Verwendung in wissenschaftlichen Arbeiten sind alle Daten wichtig, die für ein korrektes Referenzieren benötigt werden. Die im Punkt 4.2.1.2.1 („Strukturinformation“) aufgezählten Daten reichen für diesen Zweck nicht aus!

4.2.1.2.4 Beziehungen zwischen Wissenseinheiten

Eingeschränkt können auch Referenzen (Beziehungen) zwischen Wissenseinheiten automatisch erkannt werden. Dies ist jedoch nur dann möglich, wenn im Fließtext **explizit ein Querverweis** zu anderen Textstellen angeführt ist. Der größte Teil dieser Metadaten muss manuell bearbeitet werden. Folgende Daten sind zu speichern:

- Art der Referenz
- Referenziertes Objekt, in diesem Fall eine andere Wissenseinheit

4.2.1.2.5 Technische Information

Die technische Information definiert technische Aspekte zur Wissenseinheit und ihrer Speicherung im Dateisystem.

- Format der Einheit (MIME-Typ)
- Größe in Bytes
- Lokaler Speicherort (in Form eines Dateipfades oder einer URL)
- Voraussetzungen zur Verwendung der Wissenseinheit (lediglich in technischer Hinsicht, z.B. installierte Programme). Gespeichert werden müssen dabei der Typ (z.B. Betriebssystem, Browser), die Bezeichnung (der Name, z.B. „Microsoft Internet Explorer 5.5“), die Versionsnummer, Installationshinweise, andere Anforderungen und gegebenenfalls die Abspieldauer.

4.2.1.3 Manuelle Nachbearbeitung der Zerlegung des Dokuments

Beim Durchführen des zuvor beschriebenen Zerlegungs- und Aufbereitungsvorganges wird noch keine Interaktion mit dem Benutzer benötigt. Anschließend an die automati-

sierte Zerlegung und Aufbereitung wird der produzierte Output dem **Benutzer präsentiert**. Es hat dann einerseits eine **manuelle Überarbeitung** der Einteilung in Wissenseinheiten, sowie andererseits eine Überprüfung der automatisch zugeordneten Metadaten zu erfolgen.

Um eine hohe Wiederverwendbarkeit und Konsistenz der Wissensbasis zu erreichen, ist es unerlässlich, dass die manuelle Nachbearbeitung von einem **Experten** auf jenem Fachgebiet, mit dem sich das Quelldokument beschäftigt, durchgeführt wird. Folgender Aspekt sei noch einmal betont: Von ausschlaggebender Bedeutung sowohl für die **Wiederverwendbarkeit** der Wissenseinheiten als auch für die **Qualität** der später auf Grundlage der Wissensbasis zu produzierenden Materialien ist die Sorgfalt, mit der bei der Bearbeitung der Metadaten vorgegangen wird [Dahn, 2001b]!

Leider ist es bis heute nicht möglich, ein Werkzeug zur Erkennung von semantischen Zusammenhängen von Texten so exakt zu implementieren, dass eine manuelle Bearbeitung überflüssig würde. Wünschenswert wäre ein Konzept oder ein Algorithmus, mit dessen Hilfe der enorme Aufwand zur Zerlegung und Aufbereitung eines Dokuments von einem Programm übernommen werden könnte.

Die manuelle Bearbeitung der Zerlegung ist unter Erfüllung der folgenden zwei Voraussetzungen effektiv möglich: der erzeugte Dateibaum muss an der Benutzungsschnittstelle übersichtlich visualisiert werden und Funktionen zur Bearbeitung der Grenzen zwischen Wissenseinheiten müssen zur Verfügung stehen [Dahn, 2001b].

4.2.1.3.1 Darstellung des erzeugten Dateibaums

Bei der Darstellung der automatisch erzeugten Zerlegung des Dokuments ist die **Abstraktion** vom dahinter stehenden Datenmodell von besonderer Bedeutung. Der Benutzer darf nicht mit Details über die Speicherung belastet werden [Dahn, 2001a]. Das bedeutet, die Präsentation des Dateibaums muss so erfolgen, dass ein Benutzer ohne Programmier- oder Informatikkenntnisse diese verstehen kann. Eine Konzentration auf **bestimmte Ausschnitte** des Dateibaums muss vom Benutzer konfigurierbar sein:

- Einzelne Wissenseinheiten
- Ausschnitte des Dateibaums (dabei handelt es sich um Kapitel oder Unterkapitel)
- Gesamter Dateibaum (das Quelldokument in der vorläufigen Version der aufbereiteten Form)

Folglich muss die Darstellung der vorläufig generierten Wissenseinheiten und die zwischen ihnen herrschende Hierarchie am besten mit Hilfe einer **grafischen Benutzungsschnittstelle** erfolgen. Als Möglichkeit zur Darstellung ist die Baumform (so wie z.B. im Windows Explorer) gut geeignet, aber auch andere Darstellungsformen sind denkbar. Die **Baumform** eignet sich besonders zur Darstellung der Hierarchien, die zwischen den Wissenseinheiten herrschen; also für die Darstellung von Teilen des Dateibaums oder des gesamten Dateibaums. Für die Präsentation einzelner Wissenseinheiten ist die **Tabellenform** oder eine vergleichbare Form vorzuziehen [Dahn et al., 2001].

4.2.1.3.2 Bearbeitung der Grenzen zwischen den Wissenseinheiten

Automatisch generierte Grenzen zwischen Wissenseinheiten können im Hinblick auf Verständlichkeit und semantische Vollständigkeit **Mängel aufweisen**. Das Splitter-Programm kann eine Texteinheit unterteilt haben, obwohl dadurch keine geschlossene semantische Bedeutung mehr gegeben ist. Auf der anderen Seite können Wissenseinheiten entstehen, die ohne Verlust der semantischen Vollständigkeit in zwei oder mehrere Wissenseinheiten aufgeteilt werden könnten. Folglich müssen folgende Aspekte der Bearbeitung unterstützt werden (vgl. [Dahn et al., 2001], [Dahn, 2001b]):

- **Zerteilung** von zu großen Wissenseinheiten in kleinere semantische Einheiten
- **Zusammenfügen** von zu kleinen Wissenseinheiten zu einer semantischen Einheit

Wenn das Ergebnis der automatisch generierten Zerlegung besonders stark vom gewünschten Ergebnis abweicht, ist zudem wünschenswert, dass der **Splitter** während des Schrittes der manuellen Nachbearbeitung **rekonfiguriert und nochmals gestartet** werden kann. Mit anderen Worten muss zu diesem Zeitpunkt eine Rückkopplungsschleife zum Splitter eingebaut werden, mit deren Hilfe eine wenig sinnvolle manuelle Bearbeitung vermieden werden kann.

4.2.1.4 Manuelle Bearbeitung und Ergänzung der Metadaten

Neben den Mängeln, die bei den generierten Grenzen zwischen Wissenseinheiten auftreten können, sind auch Unzulänglichkeiten bei den automatisch extrahierten und den Wissenseinheiten zugewiesenen Metadaten denkbar. Mit großer Wahrscheinlichkeit erfordert die Bearbeitung der Metadaten sogar noch signifikant mehr Aufwand seitens des Benutzers als die Bearbeitung der Zerlegung.

Wesentliche Aspekte der Bearbeitung von Metadaten sind, dass Einschränkungen der Verwendbarkeit aufgrund des **Urheberrechts** modellierbar sein müssen; die **Beschlagwortung** muss ergänzt und/oder korrigiert werden können; eventuell können sogar **pädagogische Aspekte** berücksichtigt werden. Letztlich geht es darum, dass das Wissen des Experten mit Hilfe der Metadaten in der Wissensbasis abgebildet und gespeichert werden kann [Dahn, 2001b].

Für die Bearbeitung der Metadaten kann prinzipiell die selbe Benutzungsschnittstelle wie bei der Bearbeitung der Zerlegung verwendet werden. Vom ergonomischen Gesichtspunkt aus betrachtet ist die Darstellung in einer einzigen Benutzungsschnittstelle von Vorteil, da für den Benutzer die Struktur des Dokuments von zusätzlicher Information über das Dokument (also den Metadaten) mental kaum trennbar ist.

Folgende Kategorien der Metadaten sind jene, deren Bearbeitung überwiegend oder ausschließlich durch Experten erfolgen muss (vgl. [Dahn et al., 2001], [Dahn, 2001b]):

- Aussagekräftiger **Name** der Wissenseinheit
- **Einordnung** in ein Teilgebiet des Fachgebiets
- **Sprache** der Wissenseinheit
- **Beschreibung** des Inhalts der Wissenseinheit

- **Beschlagwortung** der Wissenseinheit; diese erfolgt am besten auf der Grundlage eines **Thesaurus**. Die große Gefahr bei der Beschlagwortung ist die Verwendung einer von Individuum zu Individuum unterschiedlichen Begriffswelt; dieser Nachteil kann mit Hilfe von Thesauri in den Griff bekommen werden.
- Information über den **Lebenszyklus** einer Wissenseinheit; es werden **Versionsnummer** (in diesem Fall immer die erste), **Status** der Wissenseinheit (z.B. Entwurf, überarbeitete Version, zur Verwendung freigegeben), **bearbeitende Person(en)** und ein **Datum** der letzten Überarbeitung abgespeichert.
- **Kategorisierung**: welche Art von semantischer Einheit liegt vor? Denkbar sind z.B. Fließtext, Abbildung, Beispiel, Tabelle, ... Vgl. dazu Abschnitt 4.2.1.2.2 Kategorisierung.
- Instruktionen zur **pädagogischen Verwendung**: z.B. Grad der Interaktivität, Umfang des semantischen Inhalts, empfohlener Benutzer, empfohlenes Alter, Schwierigkeitsgrad, ...
- Information über **urheberrechtliche** Einschränkungen der Benutzung
- **Verweise** auf andere Wissenseinheiten; wobei hier eine Unterscheidung möglich sein muss zwischen Referenzen auf Voraussetzungen zur Erlernung des Wissens (so genannte **„Rückwärtsreferenzen"**), Referenzen auf gleichwertiges Wissen (**„siehe-auch"** Referenzen) und Referenzen auf Wissenseinheiten, die auf das aktuelle Wissen aufbauen (so genannte **„Vorwärtsreferenzen"**).

4.2.1.5 Tools der Slicing Book Technology

Eine Implementierung der SBT ist bereits kommerziell verfügbar. Die **Slicing Information Technology GmbH**[13] bietet fertig aufbereitete elektronische Bücher an, die mit Hilfe der SBT durch Metadaten angereichert wurden.

Die in den Abschnitten 4.2.1.1 bis 4.2.1.4 vorgestellten Konzepte wurden in vier Werkzeuge (Tools) implementiert, die zur Erstellung eines elektronischen Buchs mit Hilfe der SBT erforderlich sind: Splitter: Tool zur Zerlegung, Tool zur Speicherung der Wissenseinheiten, Knowledge Management System und Auswahl-Tool. In diesem Abschnitt werden die Werkzeuge kurz beschrieben.

4.2.1.5.1 Splitter: Tool zur Zerlegung

Der Splitter ist das zentrale Werkzeug für die Schaffung digitalisierter Bücher mit Hilfe der SBT. Aus einem elektronischen Quelldokument erzeugt der Splitter **semantische Einheiten**, wobei durch die Konfiguration von Parametern die Granularität (Größe) der erzeugten Einheiten vorgegeben werden kann.

Die **Granularität** kann zwischen einer einzelnen Zeile und einem ganzen Kapitel schwanken, wobei das Optimalmaß zwischen diesen beiden extremen Ausprägungen liegt. Die Autoren geben als Faustregel für das Optimalmaß an [Dahn, 2001a], [Dahn et

[13] Siehe http://www.slicing-infotech.de/

al., 2001]: „Ein denkbarer Kandidat für eine Wissenseinheit ist ein Teilstück eines Dokumentes, das unter wohldefinierten Bedingungen wieder verwendet werden kann." („A potential slice is a connected part of a document that can be reused under well defined conditions.") Leider lassen die Autoren offen, was unter „wohldefinierten Bedingungen" zu verstehen ist. Weder eine Präzisierung noch ein Beispiel werden genannt.

Bei der Zerlegung elektronischer Dokumente in Wissenseinheiten ist insbesondere darauf zu achten, dass eine einzelne Einheit eine in sich **geschlossene semantische Bedeutung** besitzt. Mit anderen Worten: eine Wissenseinheit sollte verstanden werden können, ohne dass zusätzliche Information benötigt wird. Wenn aus einer Wissenseinheit keine Teile der Daten so entfernt werden können, dass ein Verständnis des (semantischen) Inhalts noch möglich ist, dann wird die optimale Granularität der Wissenseinheiten erreicht.

Zweite Aufgabe des Splitters ist die Anreicherung der erzeugten Wissenseinheiten mit **Metadaten**. Soweit dies möglich ist, geschieht die Zuordnung von Metadaten zu den Wissenseinheiten automatisch. Zur Zeit kann lediglich Strukturinformation automatisiert ermittelt und zugewiesen werden.

In einem letzten Bearbeitungsschritt unterstützt der Splitter das **manuelle Nachbearbeiten** sowohl der Zerlegung des Dokuments als auch der den Wissenseinheiten zugewiesenen Metadaten. Für diesen Schritt wird empfohlen, dass er nur von einem **Experten** auf dem vom bearbeiteten Dokument behandelten Fachgebiet vorgenommen werden sollte. Grund dafür ist, dass die **Qualität der Metadaten** umso mehr steigt, je besser der Bearbeiter mit dem Fachgebiet vertraut ist. Mit der Qualität der Metadaten steigt wiederum die **Wiederverwendbarkeit**.

4.2.1.5.2 Tool zur Speicherung der Wissenseinheiten

Das Speicherungs-Tool dient dazu, einerseits die geschaffenen **Wissenseinheiten** und andererseits die dazu gehörenden **Metadaten abzuspeichern**. Wissenseinheiten werden in Form eines semantischen Netzes gespeichert. Metadaten können in einem relationalen Datenmodell gespeichert werden, d.h. in einer Tabelle.

Bei der Speicherung ist darauf zu achten, dass **Such- und Zugriffsfunktionen** möglichst effizient bereit gestellt werden. Wichtig ist vor allem, dass eine nachträgliche Bearbeitung von Wissenseinheiten möglich ist. Dies dient dazu, die Wissenseinheiten immer auf dem neuesten Stand der Forschungen halten zu können, für den Fall, dass sich Erkenntnisse ändern oder erweitern.

4.2.1.5.3 Knowledge Management System

Das Knowledge Management System (KMS) wird bei der **Erzeugung von Dokumenten** aus den Wissenseinheiten der Wissensbasis herangezogen. Mit Hilfe des KMS werden aus der Wissensbasis Vorschläge für die zu berücksichtigenden Wissenseinheiten generiert [Dahn, 2001a], [Dahn, 2000c].

Dieser Vorgang stellt eine sehr komplexe Aufgabe dar. Ausgehend von den vorhandenen Wissenseinheiten, einem Modell des Benutzers sowie bestimmten vorgegebenen Regeln muss das KMS in der Lage sein, **Schlüsse zu ziehen**, welche Wissenseinheiten in die

Auswahlmenge aufzunehmen ist. Das KMS ist also als **Expertensystem** implementiert und mit einer **Inference engine** (also einem Mechanismus, der durch Schlussregeln logische Schlüsse ziehen kann) ausgestattet.

Das Benutzermodell berücksichtigt die persönlichen Vorlieben und Interessen, den vorhandenen Wissensstand sowie die Lernziele des Benutzers.

4.2.1.5.4 Auswahl-Tool

Das Auswahl-Tool dient dazu, die vom KMS generierten Vorschläge zu überarbeiten. Es können Wissenseinheiten aus der Auswahlmenge eliminiert, zusätzliche Wissenseinheiten in die Auswahlmenge übernommen sowie die Reihenfolge der Wissenseinheiten im Dokument verändert werden [Dahn, 2001a].

4.3 Konzepte zur Speicherung zerlegter Dokumente

In diesem Abschnitt werden Konzepte aus Wissenschaft und Praxis untersucht, die sich mit der Speicherung fertig zerlegter elektronischer Dokumente beschäftigen. Gemäß der Gliederung der Literaturanalyse (siehe Abschnitt 4.1.2) werden im Folgenden behandelt:

4.3.1 IEEE Learning Object Model

Das IEEE[14] Learning Technology Standards Committee (IEEE LTSC) prägte als Grundlage für den Learning Object Metadata Standard den Begriff der „Learning Objects". Der Begriff ist wie folgt definiert. „A Learning Object is **any entity**, digital or non digital, that can be used, re-used or referenced during technology-supported learning" ([Koper, 2001], S. 4).

Wie man sieht, ist der Begriff **sehr weit und universell gezogen**; er umfasst sämtliche Objekte der Wirklichkeit, die zur Abbildung von Wissen und in Folge dessen für Lernvorgänge verwendet werden können. Learning Objects sind also Entitäten, welche zur Speicherung von Wissen als Ausschnitt der Wirklichkeit dienen.

Obwohl es bereits einen offiziell verabschiedeten Standard für Metadaten zu Learning Objects gibt (vgl. die Analyse im Abschnitt 4.3.2: IEEE Learning Object Metadata Standard, Seite 54 ff.), sind die Learning Objects selbst noch nicht strikt definiert. Allgemein anerkannte Ansicht ist jedoch, dass ein Learning Object aus Inhalt und Methoden besteht, wobei beide Bestandteile optional vorkommen, d.h. auch fehlen dürfen (vgl. [Koper, 2001] und [Downes, 2000]).

---

[14] IEEE = Akronym für Institute of Electrical and Electronics Engineers, Inc. (http://www.ieee.org). Siehe dazu auch das Glossar (Seite 333 ff.).

- **Inhalt** von Learning Objects ist so definiert, dass er aus Daten (in Form von Text, Bildern, Video-Dateien, Audio-Dateien, usw.) und (Referenzen auf) andere Learning Objects bestehen kann.
- **Methoden** sind Beschreibungen des Verhaltens eines Learning Objects.
- Zur Beschreibung von Learning Objects können **Metadaten** gemäß des Learning Object Metadata Standards (vgl. Abschnitt 4.3.2) eingesetzt werden.

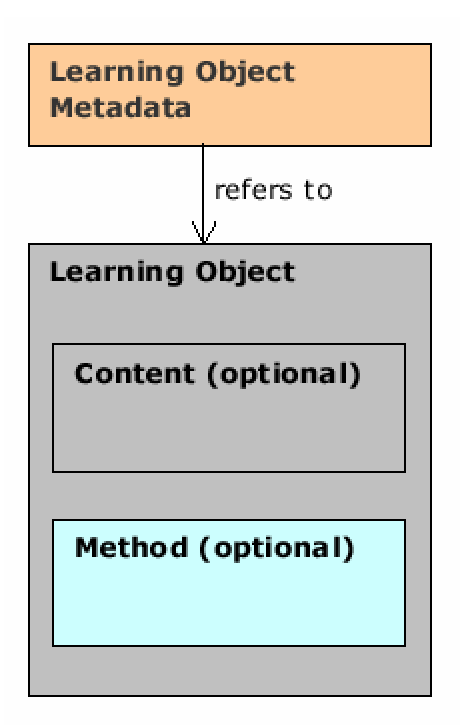


Abbildung 6: Learning Objects – konzeptionelle Sicht (nach [Koper, 2001])

Wie aus der Abbildung 6 ersichtlich wird, existieren die Metadaten unabhängig von den Learning Objects. Die Learning Objects wie auch die Metadaten dazu können in Datenbanken abgelegt und zu „Paketen" zusammengesetzt werden.

Hinweis: Das IMS Konsortium hat bereits eine Spezifikation veröffentlicht, die zur Bündelung von Learning Objects dient: die IMS Content Packaging Specification. Diese wird im Abschnitt 4.3.3.2: Lernmaterial-Struktur, Seite 63 ff. analysiert.

4.3.2 IEEE Learning Object Metadata Standard

Das IEEE Learning Object Model bzw. in diesem Zusammenhang der IEEE Learning Object Metadata (LOM) Standard ist bis jetzt auf dem Gebiet der Content- bzw. Metadaten-Verwaltung der **einzige existierende** offizielle und anerkannte Standard. (Anmerkung: zur Definition des Begriffs „Standard" siehe die Einleitung zu diesem Kapitel im Abschnitt 4.1.1: Allgemeine Terminologie auf Seite 36.)

Zweck des IEEE LOM Standards ist die **Vereinfachung der Verwaltung und Nutzung** der so genannten „Learning Objects" (zur Definition des Begriffs siehe Abschnitt 4.3.1). Dazu zählen die Suche, Entwicklung, Bewertung, Distribution und Verwendung.

Metadaten werden als **Voraussetzung** angesehen, um **Kataloge und Datenbasen** (inventories) aus Learning Objects (Wissensatomen) zu erstellen. Dies ist notwendig, um den Austausch von Wissen zu vereinfachen. Ohne eine **standardisierte Modellierung von Metadaten** zu den Wisseneinheiten kann Interoperabilität zwischen Systemen unterschiedlicher Hersteller niemals erreicht werden [LOM, 2000].

In diesem Abschnitt geht der Autor auf folgende Aspekte des Learning Objects Metadata Standards ein:

4.3.2.1 Überblick über die Struktur der Metadaten

Im Learning Object Metadata Standard werden folgende Kategorien von Metadaten gespeichert. Es werden jeweils die in der Quelle [LOM, 2000] angegebenen englischen Originalbezeichnungen angegeben und durch eine kurze Beschreibung ergänzt:

- **General**: allgemeine Information zum Learning Object (die sich auf das Objekt als Ganzes bezieht).
- **Lifecycle**: Daten, die sich auf die **Versionshistorie** und den **gegenwärtigen Status** des Learning Objects beziehen. Zudem werden alle Autoren, die das Learning Object verändert haben sowie die Zeitpunkte, an denen die Veränderungen vorgenommen wurden, gespeichert.
- **Meta-metadata**: Information über die Metadaten selbst, d.h. eine Beschreibung der Metadaten (Daten über die Metadaten).
- **Technical**: technische Anforderungen und Eigenschaften des Learning Objects werden zusammen gefasst.
- **Educational**: gibt Information über **pädagogische Aspekte** (Eigenschaften des Learning Objects und Empfehlungen für den Einsatz in der Lehre), die das Learning Object betreffen.
- **Rights**: diese Kategorie enthält Information über den oder die **Inhaber der Urheberrechte** bezüglich des Learning Objects sowie über Einschränkungen oder Bedingungen der Nutzung des Objekts.
- **Relation**: dient zur Definition der Referenzen/Beziehungen des Learning Objects zu anderen Learning Objects.
- **Annotation**: stellt Information zur Verfügung, wie das Learning Object in den **Kontext von Fachgebieten** eingeordnet werden kann sowie über den Autor der Annotationen. In dieser Kategorie können zudem **persönliche Anmerkungen** des Autors eines Learning Objects gespeichert werden.
- **Classification**: beschreibt die Kategorisierung des Learning Objects. Unter Kategorisierung ist z.B. die Definition von Arten einer semantischen Einheit für ein bestimmtes Fachgebiet zu verstehen (vgl. Abschnitt 4.2.1.2.2 Kategorisierung). Der

Standard lässt dem Benutzer für diese Kategorie der Metadaten weitreichende Freiheiten in Bezug auf das Kategorisierungsschema, d.h. die Definition der möglichen Kategorien, die hier als Skala dienen.

4.3.2.2 Datenelemente

Die so genannten „Datenelemente" sind die **grundlegenden Bestandteile** des IEEE LOM Standards. Der Begriff ist nicht zu verwechseln mit Elementen einer XML-Datei, obwohl die beiden Konzepte sehr eng miteinander verwandt sind. Im Sinne des IEEE LOM Standards ist ein Datenelement die **Modellierung eines Aspektes der Wirklichkeit**, d.h. von Daten über ein Learning Object. Diese Modellierung geschieht unabhängig von jeder Bindung an technische Mittel für die Umsetzung oder Darstellung.

Alle Datenelemente zusammen formen das **Metadaten-Modell**. Dabei werden die Datenelemente in Kategorien gruppiert (diese wurden eben in Abschnitt 4.3.2.1 vorgestellt).

Der IEEE LOM Standard definiert für jedes Datenelement die folgenden Angaben [LOM, 2000]:

- **Name**: die eindeutige Bezeichnung in Form einer Zeichenkette, durch die das Datenelement identifiziert werden kann.
- **Explanation (Erläuterung)**: eine beschreibende Definition des Datenelements.
- **Size (Größe)**: die Anzahl der erlaubten Werte.
- **Order (Reihenfolge)**: gibt an, ob die Reihenfolge der Datenelemente für die Interpretation eine Rolle spielt oder diese vernachlässigt werden kann.
- **Value space (Wertebereich)**: die Menge der erlaubten Werte, die dem Datenelement zugewiesen werden dürfen. Der Wertebereich wird meist in Form eines Vokabulars angegeben.
- **Data type (Datentyp)**: eine Definition der zulässigen Ausprägungen von Werten, die dem Datenelement zugewiesen werden dürfen.
- **Example (Beispiel)**: ein Beispiel zur Illustration der Verwendung.

4.3.2.3 Details zum Datenmodell

Die Wiedergabe aller Details des Datenmodells zum LOM Standard würde den Rahmen der vorliegenden Diplomarbeit sprengen. Der Verwendungszweck bzw. die Bedeutung eines jeden Datenelements sowie weitere Angaben zu Multiplizität, zulässigem Wertebereich und Beispiele für die Verwendung eines jeden Datenelements sind in der Originalquelle [LOM, 2000] nachzulesen.

◆

4.3.3 IMS Global Learning Consortium Spezifikationen

IMS (Instructional Management Systems) Global Learning Consortium, Inc.[15] ist ein globales Konsortium, dessen Mitglieder zahlreiche Organisationen aus dem **universitären, kommerziellen oder staatlichen Bereich** sind. Gegründet wurde es 1997 durch die amerikanische Organisation „National Learning Infrastructure Initiative" (NLII)[16], welche wiederum durch EduCause finanziert wurde. EduCause ist eine private Non-Profit Organisation, deren Ziel darin besteht, durch die intelligente Nutzung moderner Informationstechnologie Lernvorgänge effizienter zu gestalten[17]. In der Zwischenzeit hat sich IMS zu einem unabhängigen, nicht gewinnorientierten Konsortium entwickelt, das seine Aktivitäten aus den jährlichen Beiträgen seiner Voll-Mitglieder finanziert.

Das IMS Konsortium bildet zahlreiche Arbeitsgruppen und veröffentlicht Spezifikationen zur **Modellierung von Daten**, welche von Wissenstransfer-Umgebungen benötigt werden. Dazu zählen (unter Anderem) Lehr- und Lernmaterialien, Aufzeichnungen über Lernfortschritte von Benutzern und der Austausch von Daten zur Administration eines Wissenstransfer-Systems [IMS General, 2002].

Zweck der Bemühungen des Konsortiums ist letztlich die Schaffung von weltweit anerkannten **Spezifikationen**. Durch Vereinheitlichung der Formate für den Austausch von Daten sollen die Voraussetzungen geschaffen werden, um **größt mögliche Interoperabilität** zwischen den Komponenten von Wissenstransfer-Anwendungen bzw. zwischen Anwendungen verschiedener Hersteller zu erreichen.

Ziele des IMS Global Learning Consortiums sind folglich in erster Linie:

- Technische **Standards** sollen definiert werden, um so Interoperabilität zwischen Wissenstransfer-Anwendungen und -diensten unterschiedlicher Anbieter zu ermöglichen. Auf die Vorteile bei der Verwendung von Standards wurde bereits im Abschnitt 4.1.1 hingewiesen (vgl. Seite 36 ff.).
- Die **Einbindung** der vom IMS verabschiedeten Spezifikationen in Produkte und Dienstleistungen, d.h. die Arbeit der Entwickler von Wissenstransfer-Umgebungen, soll weltweit bestmöglich unterstützt werden.

Mittlerweile wurden von IMS zahlreiche Spezifikationen veröffentlicht, mit denen alle wichtigen Aspekte und Daten einer Wissenstransfer-Umgebung modelliert werden können. Die zur Zeit unterstützten Spezifikationen umfassen Datenmodelle für folgende Bereiche:

- Metadaten (vgl. Abschnitt 4.3.3.1)
- Lernmaterial-Struktur (vgl. Abschnitt 4.3.3.2)

[15] IMS = Instructional Management Systems. Die ausführliche Schreibweise wird jedoch vom Konsortium nicht mehr verwendet, um Irreführungen durch die Bezeichnung zu vermeiden. Es wird empfohlen, lediglich die Abkürzung IMS zu verwenden. (Vgl. [IMS General, 2002], siehe auch Kapitel 10.2 Glossar, Seite 333 ff.)

[16] Siehe http://www.educause.edu/nlii

[17] Vgl. http://www.educause.edu/about/

- Modellierung von Lernenden
- Assessment, Benotung und Lernfortschritte
- Organisationsmodellierung

In der Folge werden die für die beiden erst genannten Bereiche relevanten Datenmodelle (d.h. die Spezifikationen) detaillierter behandelt. Die Datenmodelle der drei letzt genannten Spezifikationen sind für die Diplomarbeit nicht relevant. Auf eine ausführliche Darstellung wird daher verzichtet. Der interessierte Leser wird auf die Quellen [IMS Learner, 2001] (Modellierung von Lernenden), [IMS QTI, 2002] (Assessment, Benotung und Lernfortschritte), [IMS Ent XML, 2002], [IMS Ent, 2002] und [IMS Ent XML, 2002] (Organisationsmodellierung) verwiesen.

4.3.3.1 Metadaten

"IMS Learning Resource Metadata Information Model" [IMS Meta, 2001].

Die IMS Spezifikation zur Modellierung von Metadaten ist vollständig kompatibel zum **IEEE Learning Object Metadata Standard** (vgl. [LOM, 2000]). Die Spezifikation beschreibt Namen, Definitionen (d.h. Erläuterungen des Verwendungszwecks), Struktur („organisation") und Einschränkungen der Metadaten-Elemente. Folgende Kategorien (d.h. Hauptelemente) können mit Hilfe der IMS Metadaten Spezifikation abgebildet werden:

- Allgemeine Information („general") – siehe Abbildung 7
- Versionsnummerierung („lifecycle") – siehe Abbildung 8
- Meta-Metadaten („meta-metadata") – siehe Abbildung 9
- Technische Information („technical") – siehe Abbildung 10
- Pädagogische Anweisungen („educational") – siehe Abbildung 11
- Urheberrechtliche Aspekte („rights") – siehe Abbildung 12
- Referenzen auf andere Wissenseinheiten („relation") – siehe Abbildung 13
- Anmerkungen des Autors („annotation") – siehe Abbildung 14
- Kategorisierung der Wissenseinheit („classification") – siehe Abbildung 15

In den folgenden Abbildungen werden die genannten Elemente des Metadaten-Modells im Detail grafisch dargestellt. Daraus werden insbesondere die wichtigsten Subelemente ersichtlich.

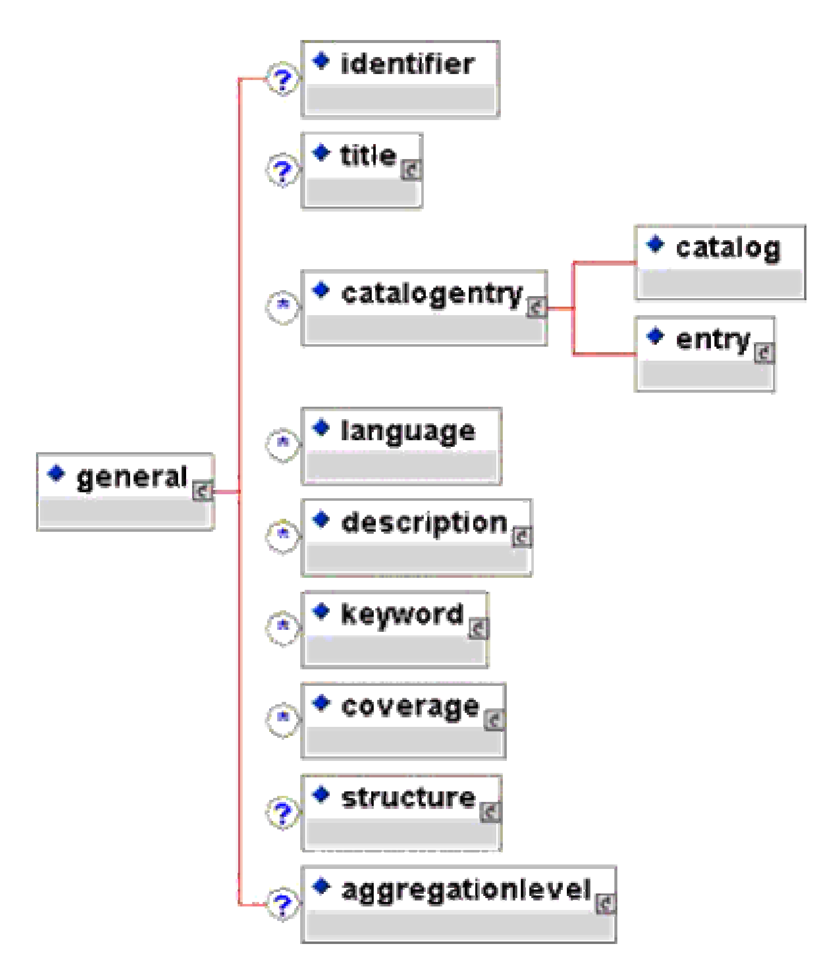


Abbildung 7: „General" Element [IMS Meta XML, 2001]

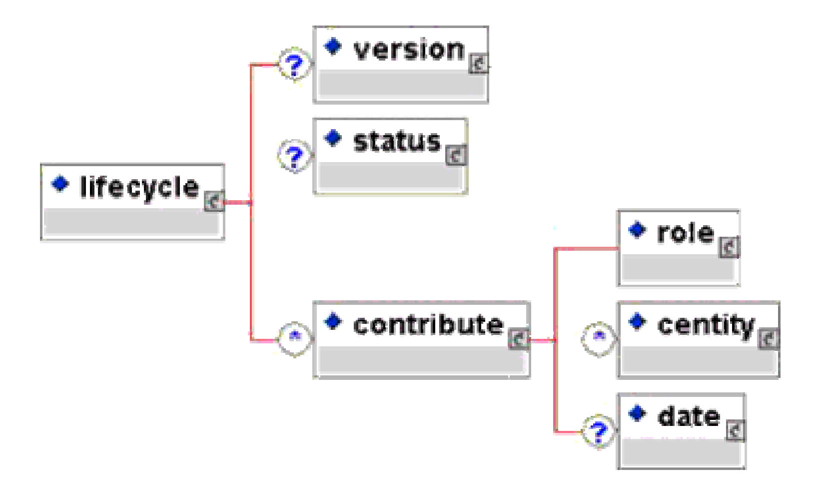


Abbildung 8: „Lifecycle" Element [IMS Meta XML, 2001]

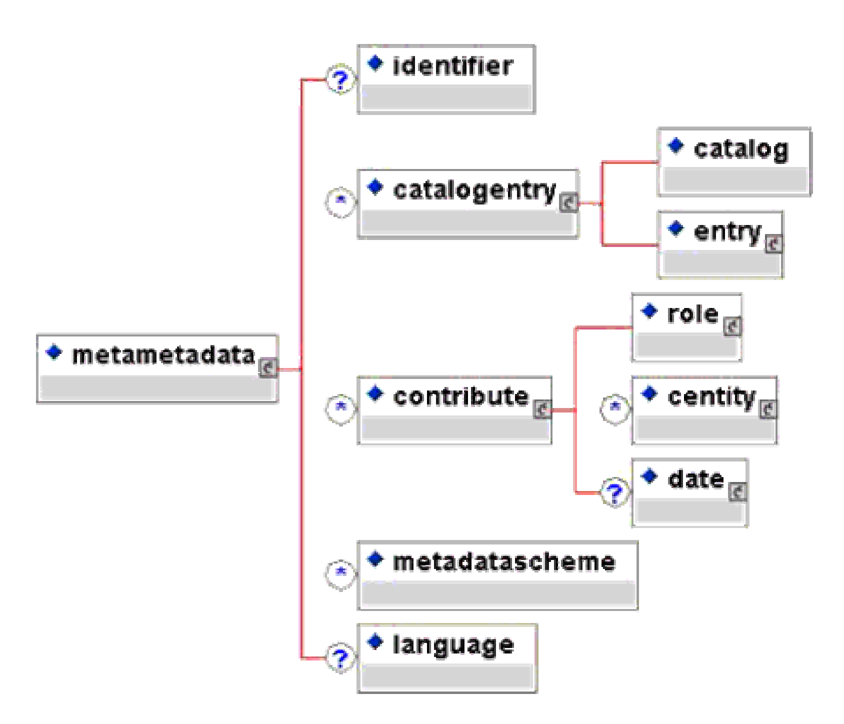


Abbildung 9: „Meta-metadata" Element [IMS Meta XML, 2001]

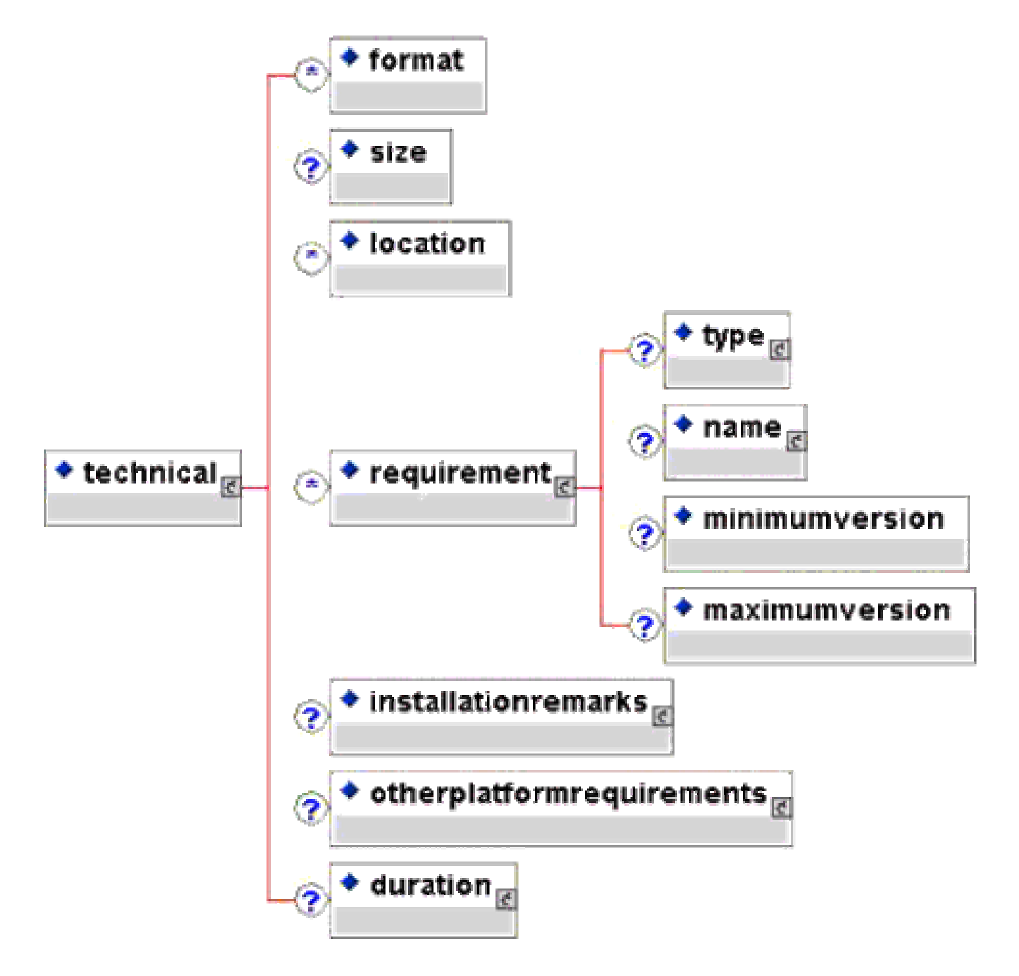


Abbildung 10: „Technical" Element [IMS Meta XML, 2001]

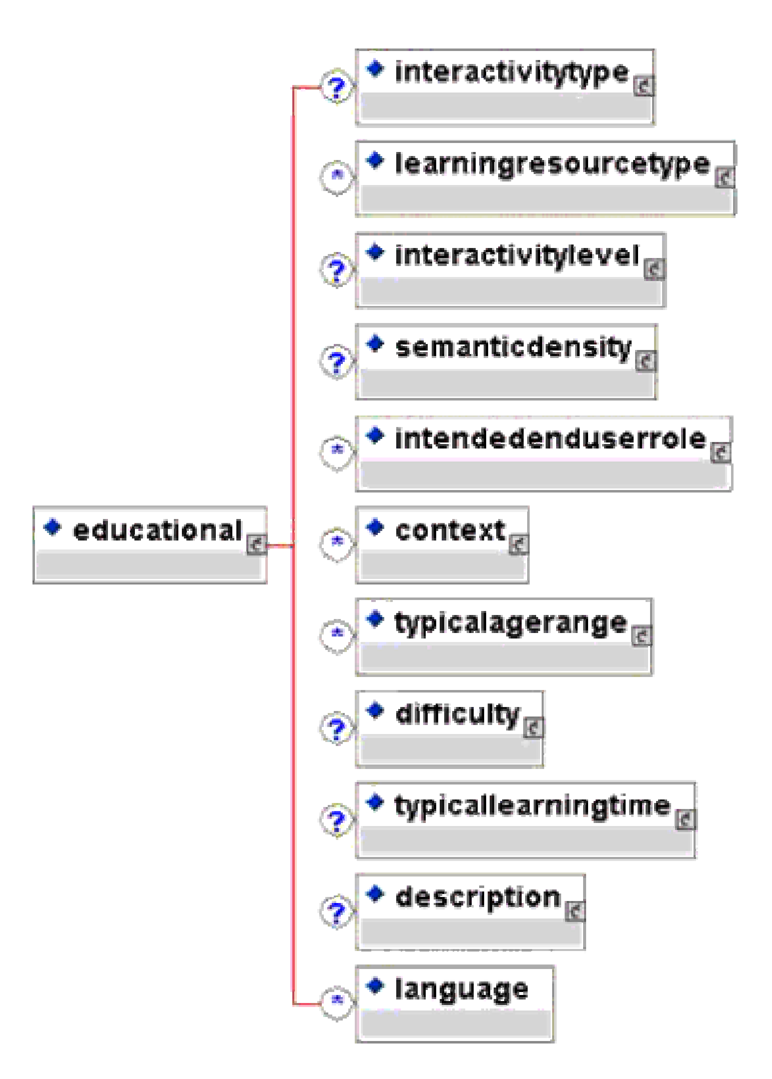


Abbildung 11: „Educational" Element [IMS Meta XML, 2001]



Abbildung 12: „Rights" Element [IMS Meta XML, 2001]

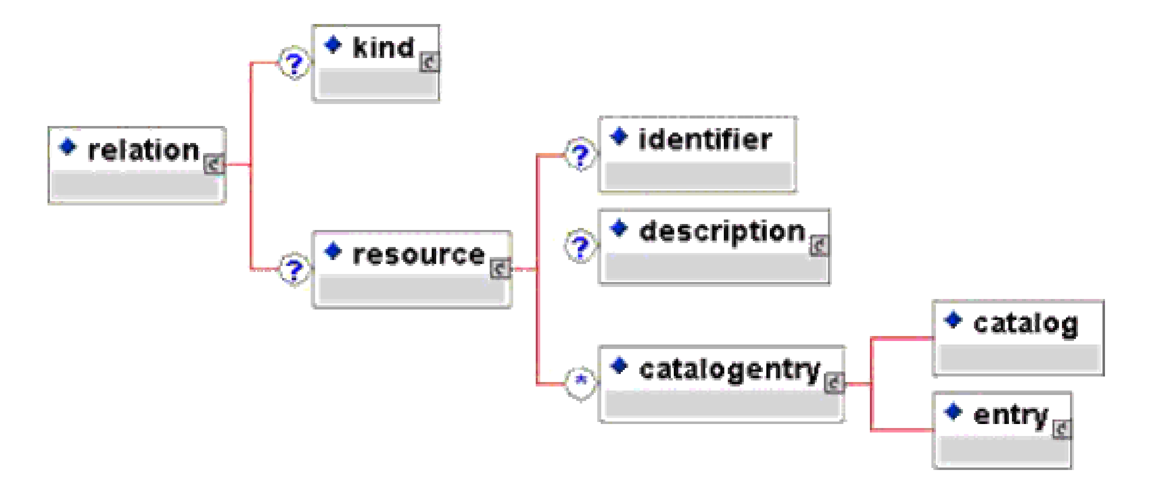


Abbildung 13: „Relation" Element [IMS Meta XML, 2001]

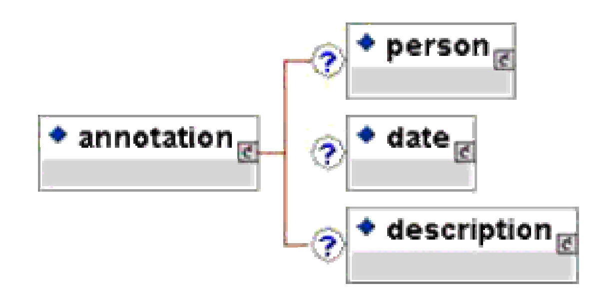


Abbildung 14: „Annotation" Element [IMS Meta XML, 2001]

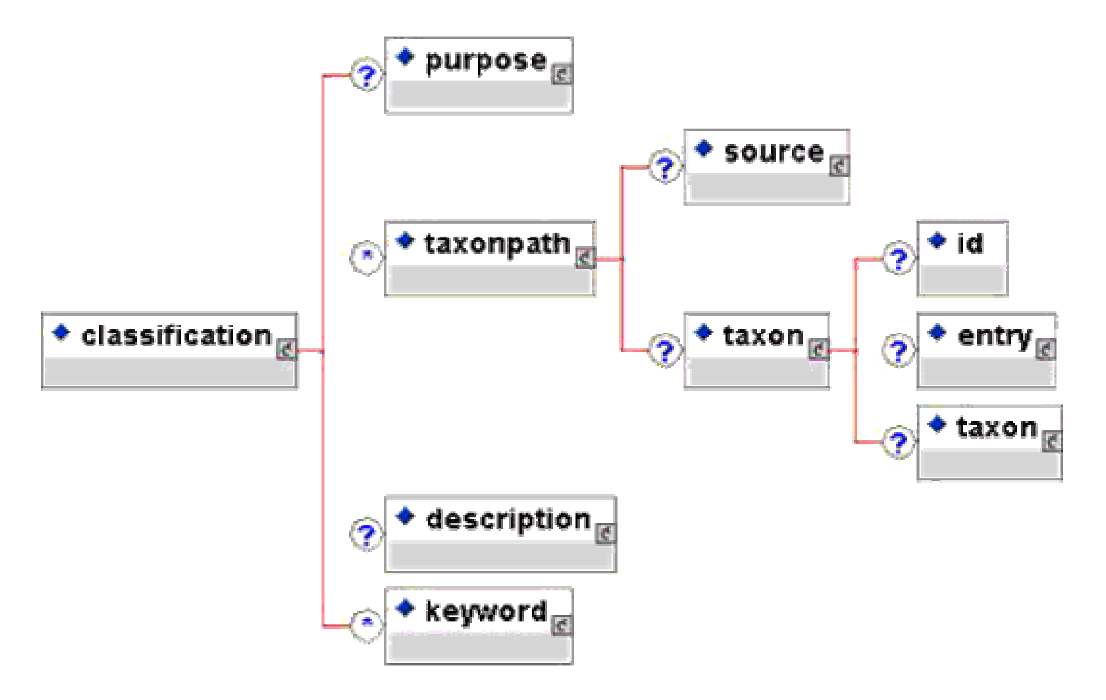


Abbildung 15: „Classification" Element [IMS Meta XML, 2001]

Für weitere Details zu den genannten Metadaten-Kategorien werden die Quellen [IMS Meta, 2001] und [IMS Meta XML, 2001] empfohlen.

4.3.3.2 Lernmaterial-Struktur

"IMS Content Packaging Information Model" [IMS Content, 2001].

Die IMS Content Packaging Spezifikation vereinheitlicht die Struktur aller Arten von **Content-Paketen**, die zur Distribution über das Internet bestimmt sind. Sie definiert für diesen Zweck eine Datenstruktur, mit deren Hilfe **digitale Inhalte aller Art** in einem so genannten „content package" zusammen gefasst und beschrieben werden können. Es ist dadurch möglich, Interoperabilität beim Importieren, Exportieren, Zusammensetzen und Zerteilen von Content-Paketen zu erreichen.

Jedes nach der IMS Spezifikation erstellte Content-Paket wird in einer Datei (z.B. einer Zipdatei) abgespeichert, die folgende Elemente enthält:

- Eine spezielle XML-Datei, genannt **Manifest**, die den Umfang des Paketes beschreibt, d.h. einerseits die Struktur („organization") und andererseits die digitalen Ressourcen (Inhalte, „resources"). Diese besteht aus vier Elementen, die weiter unten beschrieben sind.
- Alle **Ressourcen** (Dateien) in binärer Form, die durch die Manifest-Datei beschrieben werden.

Abbildung 16 (siehe weiter unten) stellt die oben beschriebenen Elemente in Form einer Übersicht dar. Die Abbildung 17 illustriert im Detail, wie die Ressourcen durch Strukturbeschreibungen („organization" Elemente) organisiert werden und auf die Ressourcen mit Hilfe der Manifest-Datei zugegriffen werden kann.

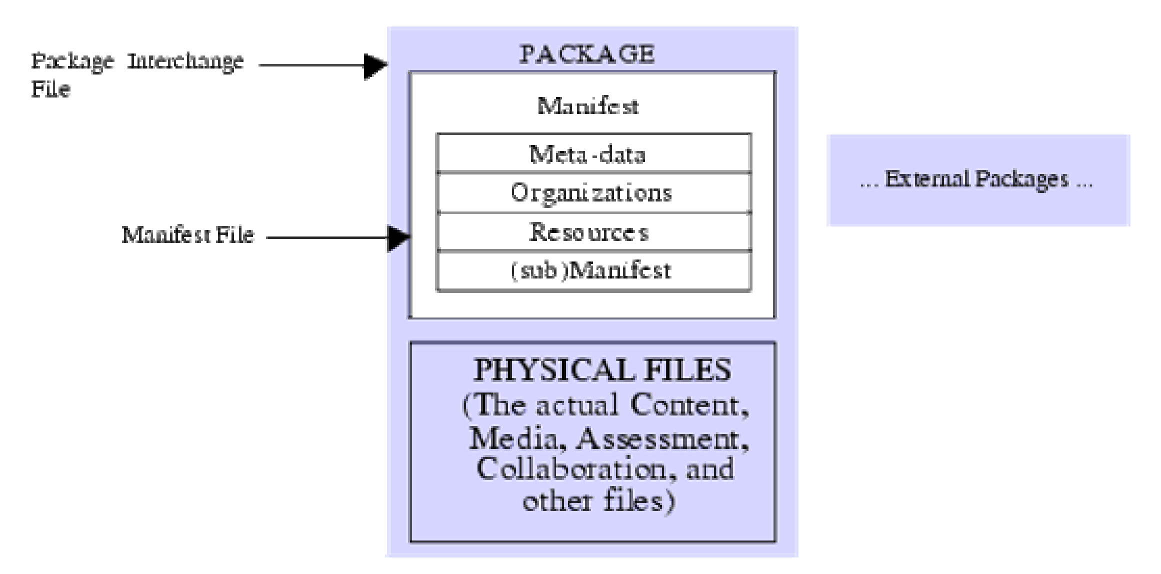


Abbildung 16: Konzeptuelles Modell eines IMS Content Package [IMS Content, 2001]

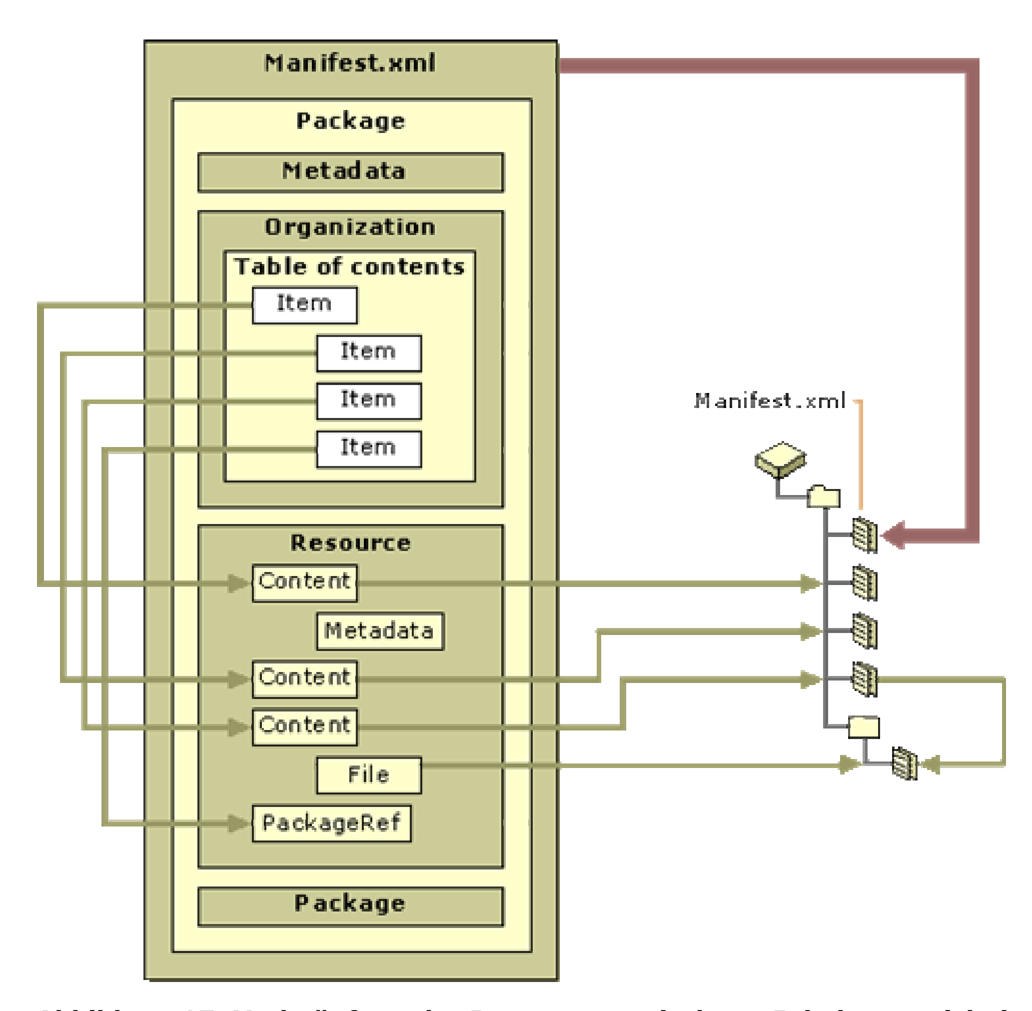


Abbildung 17: Verknüpfung der Ressourcen mit deren Inhaltsverzeichnis („organization" Element) in einem IMS Content Package (nach [Downes, 2000])

Innerhalb der Manifest-Datei werden folgende Paket-Metadaten, also Daten über die Daten im Content-Paket, gespeichert (vgl. auch die Abbildung 16):

- Ein **Metadaten-Abschnitt** („Meta-data section") enthält Information über das Manifest.
- Im Element „organizations" kann die **Struktur** beschrieben werden, die den Inhalten des Content-Paketes zugrunde liegt. Zu diesem Zweck darf das „organizations" Element ein oder mehrere „organization" Elemente enthalten. Darunter kann man Hinweise verstehen auf die **Reihenfolge**, in der die Inhalte des Pakets präsentiert oder durchgearbeitet werden sollen. Die Spezifikation mehrerer verschiedener „organization" Subelemente ist möglich. Auf diese Weise wird es möglich, speziell bei Lernmaterialien Empfehlungen für sinnvolle Lernpfade abzuspeichern und so den Lernenden einen Wegweiser durch die Daten mitzugeben.
- Das Element „resources" enthält **Verknüpfungen** zu allen physischen Ressourcen, die im Content-Paket enthalten sind. Eine Verknüpfung wird mit Hilfe eines „resource" Subelements angegeben.

- Optional ist schließlich das Element „**Submanifest**“ definiert – mit dessen Hilfe können Content-Pakete **geschachtelt** werden. D.h. dieses XML-Element erlaubt, eine Verknüpfung auf ein weiteres Content-Paket (also ein Paket im Paket) zu speichern. Eine nähere Erläuterung des Konzepts findet sich weiter unten in diesem Dokument; siehe dazu insbesondere Abbildung 19.

Die nachfolgende Abbildung 18 gibt einen Überblick (mit Kurzbeschreibungen) über die eben erwähnten Elemente.

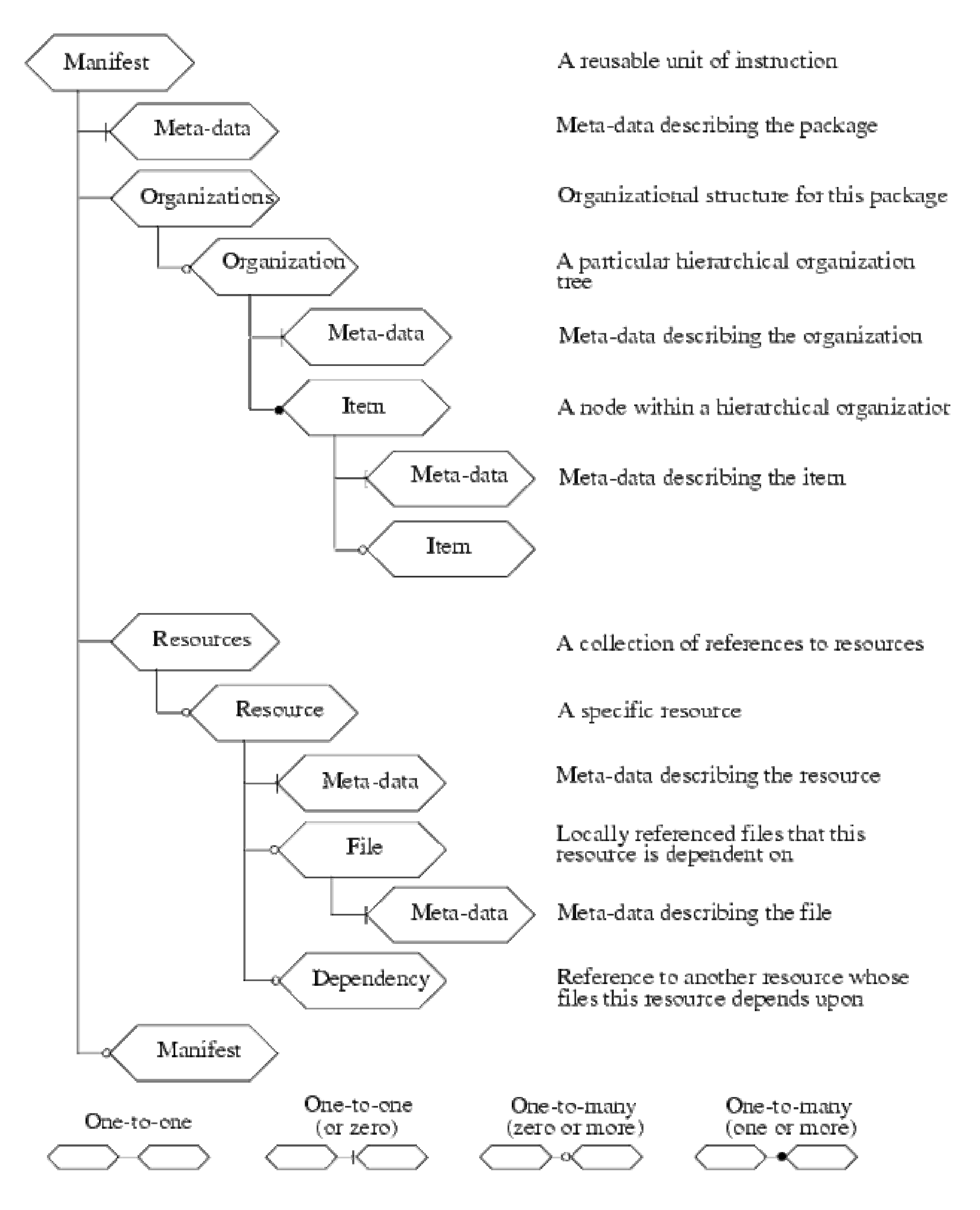


Abbildung 18: Übersicht und Beschreibung der Elemente einer Manifest-Datei [IMS Content, 2001]

Wie bereits weiter oben erwähnt wurde, besteht die Möglichkeit zur Schachtelung von Content-Packages mit Hilfe des optionalen Elements (Sub-) „Manifest“ in der XML-Manifest Datei. Abbildung 19 zeigt eine Grafik, wie dieses Konzept zu verstehen ist.

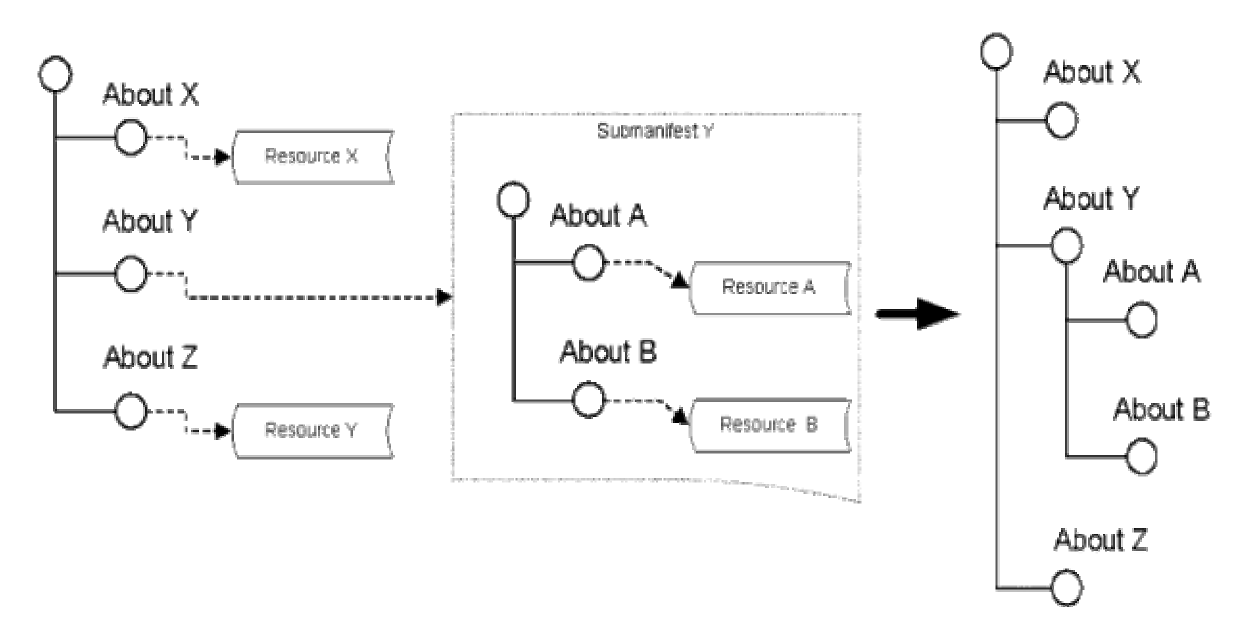


Abbildung 19: Konzept der Schachtelung von Manifests [IMS Content, 2001]

Anmerkung: die Kreise in Abbildung 19 symbolisieren **Items in einem Organization-Element** (vgl. dazu Abbildung 18). Besitzt ein Item weitere Sub-Items, so geschieht die Schachtelung wie in Abbildung 20 dargestellt.

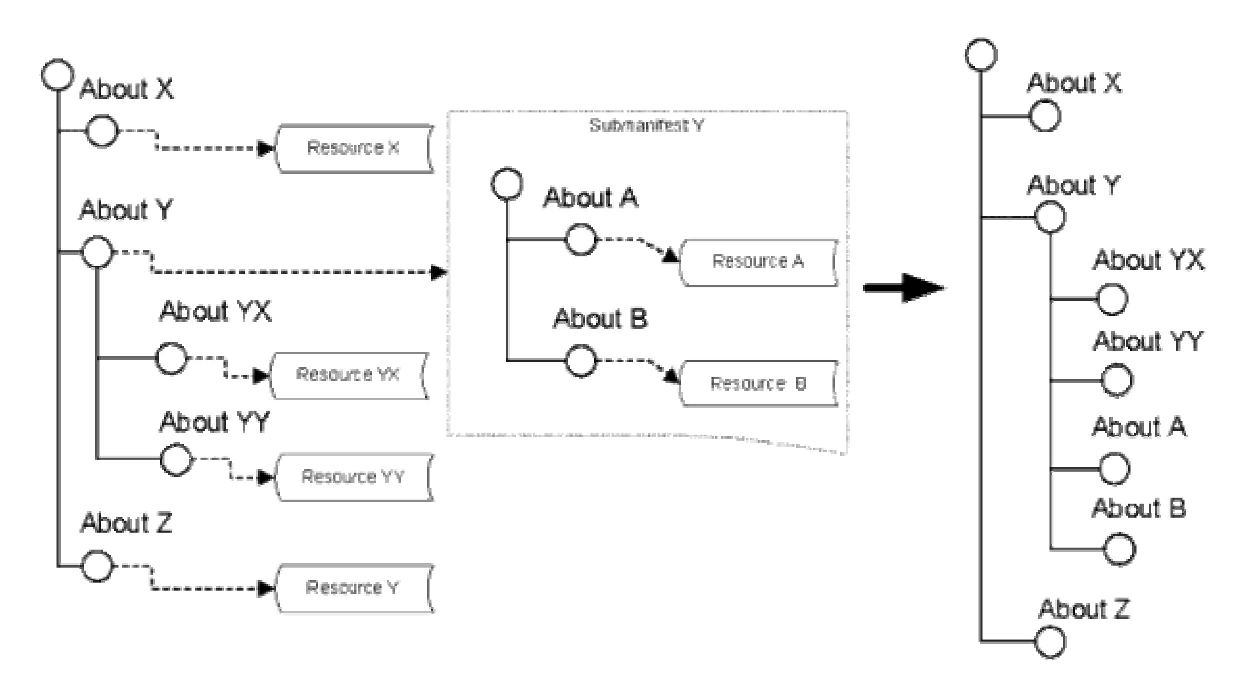


Abbildung 20: Schachtelung von Manifests unter Berücksichtigung von Sub-Items [IMS Content, 2001]

4.3.4 COMpendis Forschungsprojekt

COMpendis® ist eine Sammlung von **Werkzeugen** zur Verwaltung bzw. zum Management von Wissen. Das Paket ging aus einem Forschungsprojekt der Forschungsschwerpunkte Sprachwissenschaft und Informatik an der **Universität Salzburg** hervor. Ziel des Forschungsprojekts war die Entwicklung von Methoden, um die Unzulänglichkeiten von maschinellen Sprachverarbeitungssystemen in der Erkennung von Wortbedeutungen aufzuheben.

Laut Angaben des Herstellers, der F-Com Wissensmanagementsysteme GmbH, ist COMpendis ein Programm, das

- ... die Analyse, Klassifikation und Repräsentation von Wissen;
- ... die Strukturierung von Wissen in semantischen Netzen;
- ... die Verwaltung und Implementierung von Wissen;
- ... sowie den Erwerb und das Behalten von Wissen mit Hilfe verschiedener Lernmethoden erlaubt bzw. unterstützt [Comp, 2002].

Leider war eine exakte wissenschaftliche Analyse des Konzeptes kaum möglich. Die verfügbaren Quellen zur COMpendis Software lassen zum überwiegenden Teil wissenschaftliche Kriterien vermissen. Trotzdem konnten zumindest grundlegende Aspekte der Systemarchitektur identifiziert werden. In der Folge werden die Systemkomponenten von COMpendis vorgestellt [Comp, 2002]:

- 4.3.4.1 Semantisches Netz
- 4.3.4.2 Datenmanagement-Komponente
- 4.3.4.3 Mehrsprachiges Wörterbuch
- 4.3.4.4 Multimediale Wissensrepräsentation
- 4.3.4.5 Dokumentenmanagement

4.3.4.1 Semantisches Netz

Von zentraler Bedeutung für die COMpendis Software ist die Speicherung von Wissenseinheiten mit Hilfe von **Bedeutungsbeziehungen**. Auf diese Weise wird eine Wissensbasis auf Grundlage eines semantischen Netzes aufgebaut.

Bei COMpendis gibt es folgende drei Arten von Bedeutungsbeziehungen:

- **Typ und Untertyp**: dies entspricht einer Beziehung eines Oberbegriffs zu einem Unterbegriff.
- **Teil und Ganzes**: die so genannte Partitivrelation.
- **Thematische Relation**: Beziehung zwischen funktional assoziierten Themen.

Die Speicherung von Wissen in einem semantischen Netz garantiert eine präzise Einbettung einer Wissenseinheit in ihren fachlichen Wissenszusammenhang. Begriffliche Unschärfen, wie sie in der Fachliteratur vorkommen können, werden so vermieden [Comp, 2002].

4.3.4.2 Datenmanagement-Komponente

Für die Speicherung des semantischen Netzes in einer Datenbank ist die COMpendis Datenmanagement-Komponente verantwortlich. Diese besteht aus zwei Modulen:

- Eigentliche Wissensbasis
- Schnittstelle zum semantischen Netz

In der Folge werden diese beiden Module erläutert.

4.3.4.2.1 COMpendis Wissensbasis

Das semantische Netz von COMpendis (siehe Abschnitt 4.3.4.1) wird mit Hilfe von drei Relationen in ein relationales Datenmodell abgebildet [Grüln, 2002].

- 4.3.4.2.1.1 Relation „SEMANTISCHESNETZ"
- 4.3.4.2.1.2 Relation „WE_WSPRACH"
- 4.3.4.2.1.3 Relation "RELATION_WSPRACH"

4.3.4.2.1.1 Relation „SEMANTISCHESNETZ"

In der Relation SEMANTISCHESNETZ werden alle **Beziehungen**, die es im semantischen Netz gibt, modelliert. Dies geschieht mit Hilfe von drei Attributen:

- Attribut **„RELATION_ID"**: systemweit eindeutiger Schlüsselwert für eine Beziehung in Form einer Ganzzahl.
- Attribut **„WE_ID1"**: speichert den Schlüssel der ersten an der Beziehung beteiligten Entität. Dies entspricht einer Fremdschlüsselbeziehung zur Relation WE_SPRACH.
- Attribut **„WE_ID2"** enthält den Schlüssel der zweiten an der Beziehung beteiligten Entität. Auch dies entspricht einer Fremdschlüsselbeziehung zur Relation WE_SPRACH.

Die beiden Attribute WE_ID1 und WE_ID2 werden in Form einer Ganzzahl gespeichert. Die durch diese identifizierten Entitäten (Wissenseinheiten) stehen zueinander in semantischer Bedeutungsbeziehung.

4.3.4.2.1.2 Relation „WE_WSPRACH"

Die Relation WE_WSPRACH enthält die **eigentlichen Wissenseinheiten** in einer Tabelle, wobei eine Speicherung in unterschiedlichen Sprachen unterstützt wird. Dies geschieht mit Hilfe von drei Attributen:

- Attribut **„WE_ID"**: systemweit eindeutiger Schlüsselwert für eine Wissenseinheit in Form einer Ganzzahl. Dieses Attribut wird von den Attributen WE_ID1 und WE_ID2 in der Relation „SEMANTISCHESNETZ" (siehe Punkt 4.3.4.2.1.1) referenziert.
- Attribut **„SPRACH_ID"**: systemweit eindeutiger Schlüsselwert in Form einer Ganzzahl für eine Sprache. Es ist davon auszugehen, dass für eine Abbildung der Sprach-IDs auf Zeichenketten (welche die Sprache bezeichnen) eine weitere Rela-

tion innerhalb von COMpendis existiert. Auf diesen Aspekt geht jedoch [Grüln, 2002] nicht näher ein.

- Attribut **„WEBEZ"**: eine Zeichenkette, die die Wisseneinheit beschreibt. Die Zeichenkette modelliert die eigentlichen Wissensdaten. Somit ist in diesem Attribut der überwiegende Teil der semantischen Bedeutung abgebildet.

4.3.4.2.1.3 Relation "RELATION_WSPRACH"

Die Relation RELATION_WSPRACH übernimmt die Speicherung von **weiterer Information** zu den Beziehungen, die in der Relation „SEMANTISCHESNETZ" (siehe Punkt 4.3.4.2.1.1) abgebildet sind. Dies geschieht mit Hilfe von vier Attributen:

- Attribut **„RELATION_ID"**: systemweit eindeutiger Schlüsselwert für eine Beziehung in Form einer Ganzzahl. Es liegt eine Fremdschlüsselbeziehung zum Attribut RELATION_ID in der Relation „SEMANTISCHESNETZ" vor.
- Attribut **„SPRACH_ID"**: systemweit eindeutiger Schlüsselwert in Form einer Ganzzahl für eine Sprache.
- Attribut **„RELATIONBEZ"**: eine Zeichenkette, die erlaubt, eine textuelle Bezeichnung für die Beziehung in Richtung von Entität 1 zu Entität 2 zu vergeben.
- Attribut **„UMKEHRRELBEZ"**: eine Zeichenkette, die erlaubt, für die umgekehrte Beziehung (in Richtung von Entität 2 zu Entität 1) eine textuelle Bezeichnung zu vergeben.

4.3.4.2.2 Schnittstelle zum semantischen Netz

COMpendis unterstützt die **Eingabe** neuer Wisseneinheiten direkt vom semantischen Netz aus. D.h. während der Datenbankbetreuer sich die Darstellung des Netzes anzeigen lässt, kann er neue Wisseneinheiten in die Datenbank aufnehmen.

Durch dieses Vorgehen wird der Vorgang der Dateneingabe erleichtert. Wichtigster Vorteil ist, dass die exakte Strukturierung und Einordnung der Wisseneinheit bereits zum Zeitpunkt der Eingabe sicher gestellt wird [Comp, 2002].

4.3.4.3 Mehrsprachiges Wörterbuch

Mit Hilfe der im semantischen Netz von COMpendis gespeicherten Daten ist es möglich, Wissen auch in mehreren Sprachen zu erfassen. So wird Mehrsprachigkeit unterstützt (vgl. dazu auch Abschnitt 4.3.4.2.1 über die COMpendis Wissensbasis). Unter anderem kann die Eigenschaft der Mehrsprachigkeit dazu verwendet werden, zu einer gewünschten Sprache und einem gewünschten Ausschnitt aus der Wissensbasis ein Wörterbuch zu generieren. Folgende Ausschnitte aus der Wissensbasis können gewählt werden:

- Ein **gesamter** Fachbereich (z.B. Mathematik)
- Lediglich **ein Aspekt** eines Fachbereiches

Ob auch die Auswahl mehrerer Aspekte eines Fachbereiches möglich ist, geht aus der Quelle [Comp, 2002] nicht hervor. Ausgangspunkt für die Erstellung des Wörterbuchs ist in jedem Fall die Wisseneinheit.

4.3.4.4 Multimediale Wissensrepräsentation

COMpendis bietet die Möglichkeit, zu jeder Wissenseinheit multimediale Repräsentationen zu speichern. Einem Wissenselement können folgende multimediale Elemente zugeordnet werden [Comp, 2002]:

- Bilder
- Audiodokumente (Tondokumente)
- Animationen
- Definitionen
- Fachtexte

Die obige Aufzählung ist taxativ. Zu beachten ist, dass auch hier in Bezug auf die letztgenannten Punkte eine Zuordnung in mehreren Sprachen unterstützt wird.

4.3.4.5 Dokumentenmanagement

Mit Dokumentenmanagement ist im Sinne der COMpendis Tools die Möglichkeit zur **Speicherung eigener Dokumente** gemeint, die mit einer Wissenseinheit aus der Wissensbasis in einem thematischen Zusammenhang stehen. Es wird jedoch dem Benutzer überlassen, die semantische Zuordnung auf eine Wissenseinheit vorzunehmen [Comp, 2002].

Persönliche Dokumente können somit einem bestimmten **Begriff** (der in Form einer Wissenseinheit vorhanden ist) **zugeordnet** werden, womit die Dokumente einerseits leichter auffindbar werden und andererseits das dazu gehörende fachspezifische Wissen jederzeit abrufbar ist.

4.4 Darstellung und Verwendung zerlegter Dokumente

In diesem Abschnitt werden Konzepte aus Wissenschaft und Praxis untersucht, die sich mit der Darstellung und Verwendung fertig zerlegter elektronischer Dokumente beschäftigen. Gemäß der Gliederung der Literaturanalyse (siehe Abschnitt 4.1.2) werden im Folgenden behandelt:

4.4.1 Educational Modeling Language (EML)

Die Educational Modeling Language (EML) ist ein Vorschlag zur Standardisierung der **Beschreibung von Wissenseinheiten** im Sinn dieses Dokuments. In der Originalquelle [Koper, 2001] werden ein aus einem Contentpool erzeugtes Dokument als „unit of study", die Wissenseinheiten in der Datenbasis eines Contentpools als „Learning Objects" bezeichnet.

Grundlage des EML Standards ist das so genannte **„Learning Objects Model"**. Sinngemäß übersetzt bedeutet das ein Objektmodell für Lernmaterialien, wobei unter dem Beg-

riff „Lernmaterialien" jene Dokumente zu verstehen sind, die aus den Wissenseinheiten in einem Contentpool generiert werden. Dieses Objektmodell wird in ein **XML Schema** abgebildet, mit dessen Hilfe sowohl die Wissenseinheit selbst als auch die Metadaten dazu abgebildet werden können. Abbildung 21 zeigt eine Übersicht, wie der Autor das EML Modell illustriert (entnommen aus [Koper, 2000], S. 30).

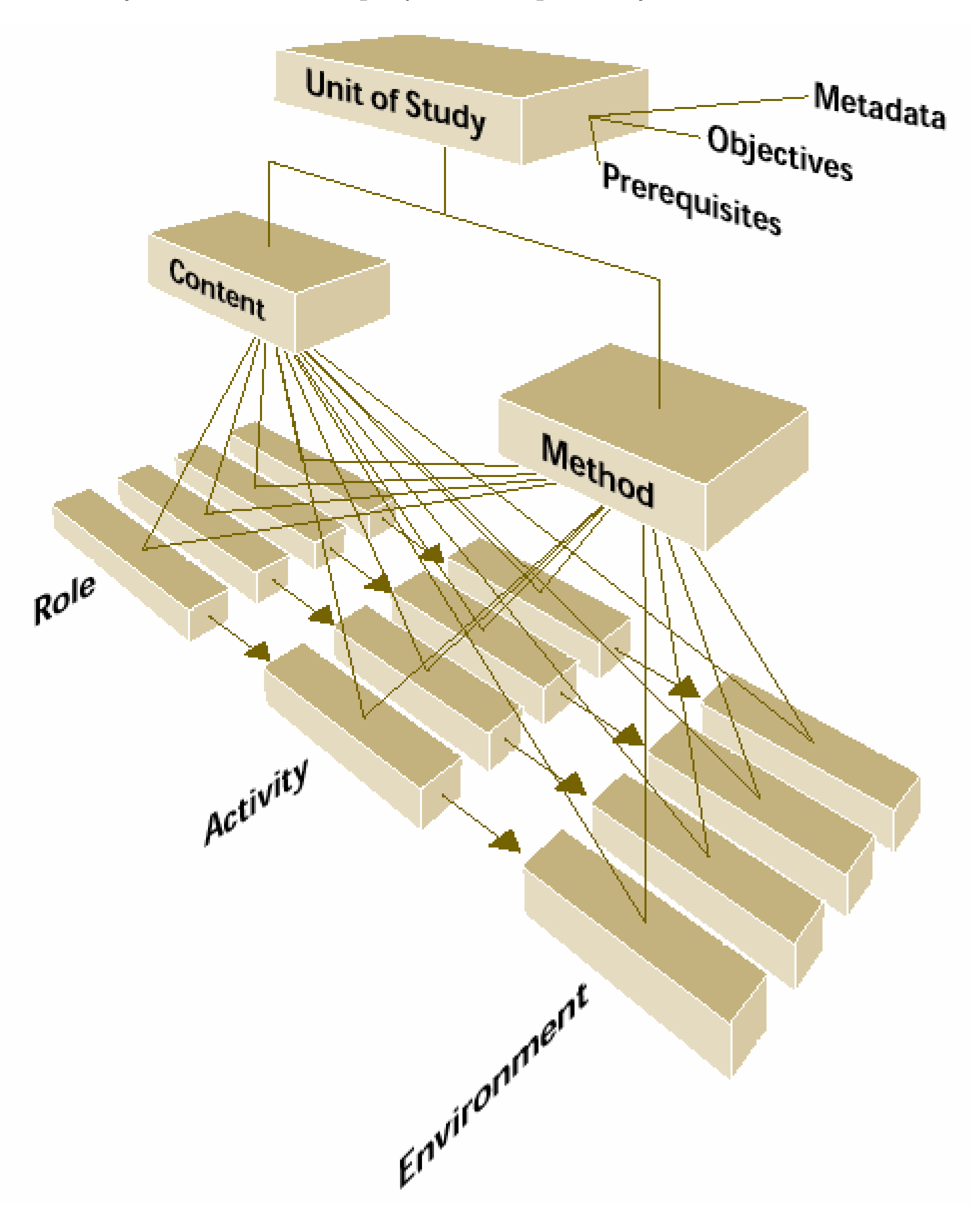


Abbildung 21: Grundlegende Struktur von EML (Konzeptioneller Entwurf)

Wie aus dieser ersten Erläuterung bereits ersichtlich, handelt es sich bei EML lediglich um ein **Beschreibungsmodell** zur Abbildung von Lernmaterialien in einer Datenbasis. Tools und Features sind daher nicht analysierbar. Stattdessen werden die wichtigsten **Aspekte und Bestandteile** des Modells erläutert:

- 4.4.1.1 Definitionen des „Learning Object Model"
- 4.4.1.2 Ziele bei der Verwendung von „Learning Objects"

- 4.4.1.3 Anforderungen an Lernmaterialien
- 4.4.1.4 Pädagogisches Metamodell
- 4.4.1.5 Implementierung mit Hilfe eines XML-Schemas

4.4.1.1 Definitionen des „Learning Object Model“

Zunächst werden in dem Modell die grundlegenden **Begriffe** "Learning Object" und "Learning Object Metadata" definiert.

4.4.1.1.1 Learning Object

Hier wird die Definition des IEEE Learning Technology Standards Committee (IEEE LTSC) als Grundlage herangezogen. Der Begriff ist wie folgt definiert: „A Learning Object is **any entity**, digital or non digital, that can be used, re-used or referenced during technology-supported learning" ([Koper, 2001], S. 4). Hier wird die Definition lediglich zur Wiederholung angegeben – der Terminus „Learning Object" wurde schon im Abschnitt 4.3.1 detailliert beschrieben (siehe Seite 53 ff.).

4.4.1.1.2 Learning Objects Metadata

Metadaten zu Learning Objects sind aus zwei Komponenten bestehend definiert:

- Einer **Referenz** auf das Learning Object (gemäß der Spezifikation des **IEEE Learning Object Models**), zu dem sie gehören
- Den eigentlichen Metadaten, die der Spezifikation des **IEEE Learning Object Metadata** Standards (IEEE LOM) genügen müssen.

In der Folge wird ein Überblick über den eben erwähnten IEEE LOM Standard gegeben.

4.4.1.1.3 IEEE LOM Standard

Aufgrund der zentralen Rolle, die dieser Standard für die EML einnimmt, wird ein kurzer Überblick über Metadaten gegeben, die der Spezifikation des LOM Standards entsprechen (vgl. [LOM, 2000]). Der Standard wurde bereits im Abschnitt 4.3.2 (siehe Seite 54 ff.) beschrieben, an dieser Stelle sind jedoch nochmals die wichtigsten Elemente des Datenmodells explizit angeführt.

Im Learning Object Metadata Standard werden folgende Kategorien von Metadaten gespeichert. Es werden jeweils die in der Quelle angegebenen englischen Originalbezeichnungen angegeben und durch eine kurze Beschreibung ergänzt:

- **General**: allgemeine Information zum Learning Object (die sich auf das Objekt als Ganzes beziehen).
- **Lifecycle**: Daten, die sich auf die Versionshistorie und den gegenwärtigen Status des Learning Objects beziehen. Zudem werden alle Autoren gespeichert, die das Learning Object verändert haben.
- **Meta-metadata**: Information über die Metadaten selbst, d.h. eine Beschreibung der Metadaten.

- **Technical**: technische Anforderungen und Eigenschaften des Learning Objects werden zusammen gefasst.
- **Educational**: gibt Information über pädagogische Aspekte, die das Learning Object betreffen.
- **Rights**: diese Kategorie enthält Information über die Inhaber der Urheberrechte des Learning Objects sowie über Einschränkungen oder Bedingungen der Nutzung des Objekts.
- **Relation**: dient zur Definition der Referenzen/Beziehungen des Learning Objects zu anderen Learning Objects.
- **Annotation**: stellt Information zur Verfügung, wie das Learning Object in den Kontext von Fachgebieten eingeordnet werden kann sowie über den Autor der Annotationen.
- **Classification**: beschreibt die Kategorisierung des Learning Objects. Unter Kategorisierung ist die Definition von Arten einer semantischen Einheit für ein bestimmtes Fachgebiet zu verstehen (vgl. Abschnitt 4.2.1.2.2 Kategorisierung).

Zum Verständnis der EML sind die oben stehenden Fakten ausreichend. An dieser Stelle wird daher darauf verzichtet, nähere Angaben über die syntaktische Struktur, Begriffswelten und weitere Aspekte der LOM Spezifikation zu geben. Für Details dazu wird auf [LOM, 2000] sowie auf den Abschnitt 4.3.2 (siehe Seite 54 ff.) verwiesen.

4.4.1.2 Ziele bei der Verwendung von „Learning Objects“

Laut [Koper, 2001] hat die Erstellung von Learning Objects nur dann einen Nutzen, wenn die folgenden Ziele in Bezug auf den Lernprozess erfüllt werden können.

- **Effektivität** (effectiveness): dem Benutzer soll die Auswahl der richtigen Lernmaterialien erleichtert werden.
- **Effizienz** (efficiency): durch die Verwendung von Learning Objects soll der Lernende seine Lernziele schneller erreichen können als bisher.
- **Attraktivität** (attractiveness): Learning Objects müssen im Vergleich zu konventionellen Lernmaterialien Vorteile bieten können, um weit reichende Akzeptanz bei den Benutzern zu erzielen.
- **Verfügbarkeit** (accessability): auf Grundlage von Learning Objects erstellte Lernmaterialien müssen möglichst umfassend, am besten rund um die Uhr, für die Lernenden zugänglich sein.

Die oben genannten Ziele sind aus der Sicht von Lernenden formuliert. Selbstverständlich lassen sich dieselben Zielkategorien auch für Tutoren, Autoren, Lehrende, u.v.m. entsprechend formulieren.

4.4.1.3 Anforderungen an Lernmaterialien

Aus den in Abschnitt 4.4.1.2 angeführten Zielen leitet [Koper, 2001] eine Reihe von Anforderungen an eine Beschreibungssprache für die so genannten „units of study“ ab. Die

EML ist nach den von ihm definierten Anforderungen entworfen. Nachfolgend wird ein Überblick über die Anforderungen gegeben [Koper, 2001]:

- **Formalisierung** (formalisation): units of study müssen durch ein formalisiertes Sprachkonstrukt **beschreibbar** sein. Diese Eigenschaft ist die Voraussetzung für eine automatische Bearbeitung, so dass der Nutzen des Lernmaterials erhöht wird.
- **Pädagogische Beweglichkeit** (pedagogical flexibility): die Beschreibungssprache muss in der Lage sein, units of study unabhängig vom pädagogischen Modell (oder Paradigma), auf dessen Basis sie gebildet wurden, zu beschreiben.
- **Explizite Kategorisierung** von Learning Objects („explicitly typed Learning Objects"): die semantische Bedeutung von Learning Objects innerhalb einer unit of study muss beschreibbar sein. Zusätzlich zur Bedeutung muss auch die Modellierung der semantischen Struktur, d.h. der Hierarchie im Lernmaterial, sowohl des Inhalts als auch des Verhaltens möglich sein.
- **Vollständigkeit der Beschreibung** (completeness of description): die Beschreibungssprache muss alle denkbaren Lernmaterialien (units of study), Wissenseinheiten (Learning Objects), Referenzen/Beziehungen (references/relationship) zwischen Wissenseinheiten, Methoden (activities) und die Arbeitsergebnisse (workflow) von Lernenden und sonstigen Benutzern beschreiben können.
- **Reproduzierbarkeit** (reproducibility): die Notation eines Lernmaterials muss die mehrmalige Ausführung seiner Methoden unterstützen, und zwar so, dass dabei die selben Ergebnisse produziert werden.
- **Personalisierbarkeit** (personalization aspects): einerseits ist damit die personalisierte, d.h. individuelle Anpassung von Inhalten und Methoden gemeint. Andererseits muss die Beschreibungssprache in der Lage sein, die Anpassung von Lernmaterialien (units of study) an persönliche Vorlieben, persönliches Vorwissen, individuelle Bedürfnisse bei der Gestaltung des Unterrichts sowie Besonderheiten von situativen Umständen zu berücksichtigen.
- **Unabhängigkeit von Medien** (medium neutrality): die Notation von Lernmaterialien (units of study) muss so erfolgen, dass die Materialien in verschiedenen Dokumentformaten (z.B. über das Web, auf Papier, als elektronisches Buch, usw.) publizierbar sind.
- **Interoperabilität und Nachhaltigkeit** (interoperability and sustainability): zwischen den Standards zur Beschreibung (Notation) eines Lernmaterials und den Techniken zu ihrer Interpretation muss über **wohl definierte Schnittstellen** streng abstrahiert werden. Auf diese Weise werden die Investitionen in den Aufbau einer Wissensbasis bzw. eines Contentpools vor revolutionären technischen Neuerungen geschützt, da eine Vorwärtskompatibilität mit noch nicht absehbaren Entwicklungen erreicht wird. Konvertierungsprobleme werden so umgangen.
- **Kompatibilität** mit gebräuchlichen Standards und Spezifikationen (compatibility): die Beschreibungssprache muss sich an heute weit verbreitete Standards anpassen können.

- **Wiederverwendbarkeit** der Learning Objects (reusability): die Notation muss die **Erkennung** (identification), **Abgrenzung** (isolation), **Zerteilung** (decontextualization) und den **Austausch** (exchange) von Learning Objects ermöglichen, so dass diese in einem anderen Kontext wieder verwendet werden können.
- Unterstützung eines **Entwicklungszyklus** für Learning Objects (life cycle): die Beschreibungssprache muss die **Erzeugung** (production), **Anpassung** (mutation), **Aufbewahrung** (preservation), **Verteilung** (distribution) und die permanente **Speicherung** (archiving) von Lernmaterialien und den ihnen zugrunde liegenden Wissenseinheiten (Learning Objects) ermöglichen.

4.4.1.4 Pädagogisches Metamodell

Hinter der EML steht ein pädagogisches Metamodell, das auf Erkenntnissen der **Lernpsychologie** aufbaut. Es besteht aus vier Hauptbestandteilen. Nachfolgend werden die in der Quelle [Koper, 2001] verwendeten englischen Originalbezeichnungen mit einer kurzen Beschreibung ihrer Bedeutung angeführt:

- **Learning model** (Lerner-Modell): beschreibt den Lernvorgang von Individuen auf der Grundlage eines Konsenses aller Lerntheorien. Das Lerner-Modell fasst, anders gesagt, den kleinsten gemeinsamen Nenner der Lerntheorien zusammen.
- **Unit of study model** (Lerneinheiten-Modell): beschreibt die Vorgehensweise zur Modellierung von praktisch brauchbaren Lernmaterialien (*units of study*). Voraussetzung ist die Kenntnis des Lerner-Modells (*learning model*) und der im Unterricht eingesetzten Didaktik.
- **Domain model** (Fachgebiets-Modell): beschreibt die Art des Inhalts und dessen Struktur. Mit anderen Worten wird das Fachgebiet (*domain*) modelliert.
- **Theories of learning & instruction** (Lern- und kognitive Theorien): die aus der Literatur entnommenen Lerntheorien und kognitiven Theorien werden im pädagigischen Metamodell berücksichtigt und abgebildet.

Für die Zwecke dieses Dokuments ist der gegebene Überblick über das pädagogische Metamodell ausreichend. Weitere Details dazu können in [Koper, 2001], S. 8 ff. nachgelesen werden.

4.4.1.5 Implementierung mit Hilfe eines XML-Schemas

Auf Grundlage der zuvor beschriebenen Definitionen, Ziele und Anforderungen wurde von der Open Universiteit Nederland[18] ein XML Schema entwickelt, mit dessen Hilfe Lernmaterialien (units of study) und Wissenseinheiten (Learning Objects) konstruiert werden können.

In Abbildung 22 wird eine vereinfachte Darstellung der Bindung von EML Unit of study Objekten an XML gezeigt (entnommen aus [Koper, 2001], S. 18). In der Abbildung wurden Attribute weggelassen. Das vollständige Schema ist auf der Internetseite der Open

[18] Siehe http://eml.ou.nl

Universiteit zum Download verfügbar. Abbildung 23 zeigt im Gegensatz dazu die Struktur einer Wisseneinheit (knowledge object).

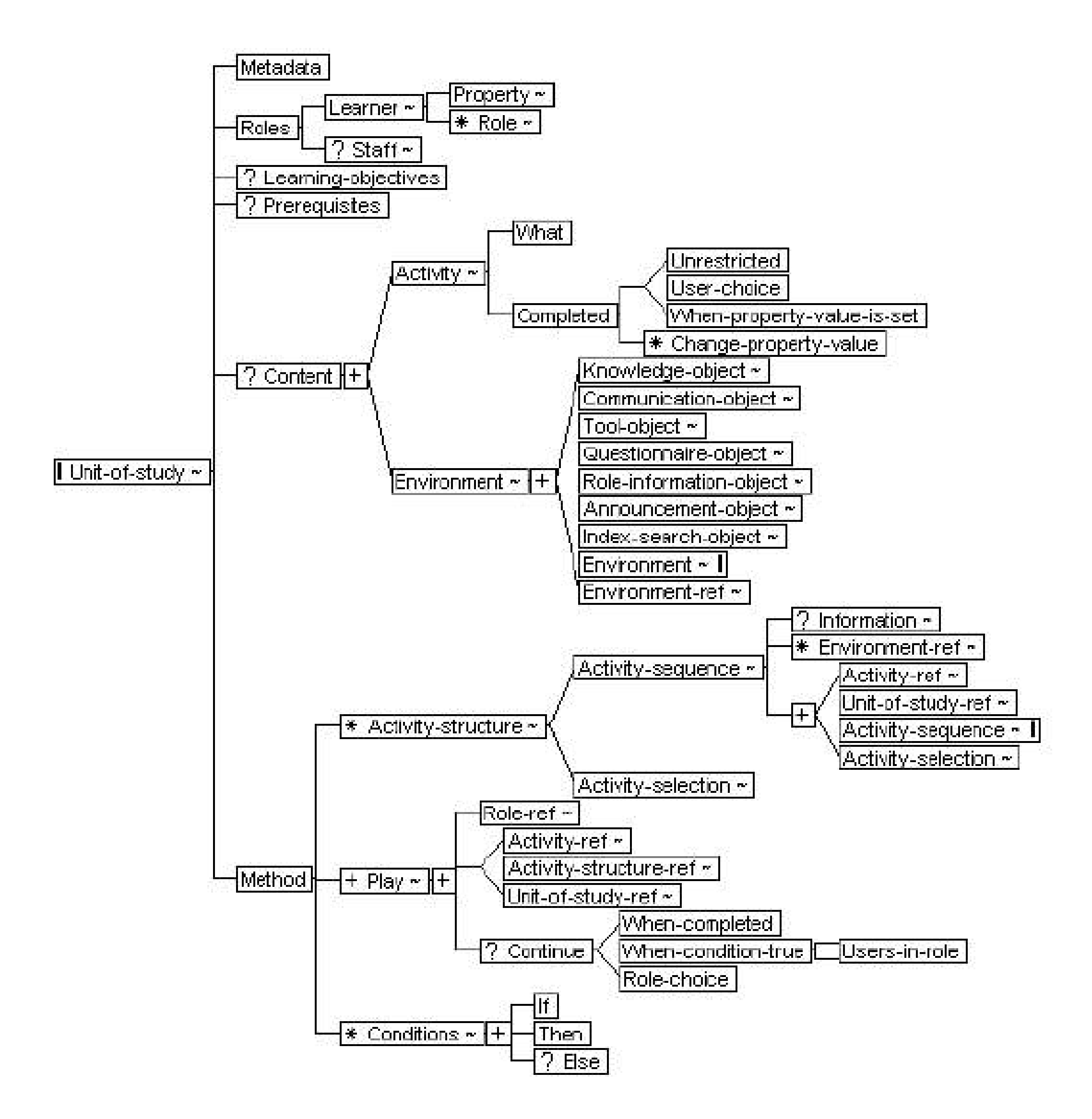


Abbildung 22: Vereinfachtes Schema für die Bindung von EML an XML (Quelle: [Koper, 2001])



Abbildung 23: Struktur einer Wissenseinheit (knowledge object) in EML (Quelle: [Koper, 2001])

Für weitere Informationen zum XML Modell wird der Leser auf die Quellenliteratur verwiesen. In [Koper, 2001], S. 20 ff. finden sich zudem Beispiele für die Formulierung von Lernmaterialien in EML.

4.4.2 XML Topic Maps (XTM)

Topic Maps sind ein Mittel zur Modellierung von Wissensbasen oder semantischen Netzen. **Semantische Netze** dienen, vereinfacht ausgedrückt, zur Abbildung von Wissen aus der Wirklichkeit. **Wissen** kann abgebildet werden, indem zusätzlich zu „nackten" **Daten** auch Information zur **Interpretation** dieser Daten gespeichert wird. Information zur Interpretation von Daten bezeichnet man als „**Semantik**".

Die grundlegende Idee bei Topic Maps ist, **externe Dokumente** mit Hilfe von **Schlagwörtern** zu referenzieren. Als Grundbestandteile des Wissens dienen die Schlagwörter (oder im Kontext von Topic Maps: die „*Topics*"). Diese werden miteinander in Verbindung

gebracht (assoziiert mit Hilfe von „*Associations*") und so der Schritt von reiner Datenspeicherung zu Wissensspeicherung vollzogen.

Wie die Formulierung „externe Dokumente" bereits andeutet, sind die Topic Maps von den referenzierten Dokumenten losgelöst – sie stellen lediglich eine **Indexstruktur des Wissens** in den Dokumenten dar. Ähnlich funktioniert ein Schlagwortindex in einem Buch – mit einem kleinen, aber wichtigen Unterschied. Ein Schlagwortindex gilt immer nur implizit; d.h. im Kontext des Buches, zu dem er gehört. Topic Maps dagegen tasten die referenzierten Dokumente nicht an. Topic Map und Dokumente können unabhängig voneinander existieren und sind zudem wechselseitig austauschbar [Widhalm, 2002].

Im Herbst 1999 wurde die Idee der Topic Maps im ISO/IEC Standard Nummer 13250 der Arbeitsgruppe ISO JTC1/SC34/WG3 offiziell standardisiert (vgl. [Widhalm, 2002], [ISO 13250/1999] und [ISO 13250/2000]). Auf Grundlage des Standards ISO 13250 wurde wenig später das unabhängige Konsortium TopicMaps.org[19] gegründet. Ziel des Konsortiums ist es, das von der ISO spezifizierte **Topicmap-Paradigma** (vgl. [ISO 13250/1999], [ISO 13250/2000]) durch eine Implementierung in XML für das World Wide Web verfügbar zu machen.

Die Spezifikation der XML Topic Maps (XTM) stellt ein Modell und eine Grammatik zur Verfügung, mit dessen Hilfe die oben beschriebenen „*Topics*", Beziehungen zwischen Topics („*Associations*") sowie so genannte „*Topic Maps*" modelliert werden können (vgl. [XTM, 2001]). Wie beim EML Konzept (vgl. Abschnitt 4.4.1, ab Seite 70) handelt es sich auch bei den XML Topic Maps um eine **Modellierungs- und Beschreibungssprache**, nicht jedoch um einen Werkzeugkasten.

Mit der Veröffentlichung der XTM Spezifikation 1.0 wurde der ISO 13250 Topic Map Standard auf die Auszeichnungssprache XML portiert. Dabei wurde auf HyTime und SGML (siehe dazu die Definitionen im Glossar, Abschnitt 10.2, Seite 333 ff.), die Grundlagen des ISO 13250 Standards, verzichtet. Statt dessen wurde auf die **Auszeichnungssprache XML** sowie die dazu gehörenden **Referenzierungsmechanismen XLink und XPointer** gesetzt (vgl. [Widhalm, 2002], S. 369).

Die beiden erwähnten Konzepte, nämlich der ISO Topic Map Standard 13250 und die XML Topic Maps, sind nicht miteinander zu verwechseln. XTM enthält im Vergleich zum Standard einige Erweiterungen. Der Rest des vorliegenden Kapitels wird sich vor allem mit dem letzt genannten Konzept, den XTM, beschäftigen. Im Vergleich zum ISO Standard sind die XML Topic Maps technisch weniger umfangreich und auch weniger aufwendig zu implementieren. Daher stellen sie die attraktivere Variante der Topic Maps dar.

Die wichtigsten **Bestandteile des ISO Topic Map Standards** sind die Konzepte „*Topic*", „*Association*" und „*Occurrence*". Im Detail besteht der Standard aus folgenden Teilkonzepten (vgl. [ISO 13250/1999], [ISO 13250/2000]):

- *Topic*
- *Topic Name*

[19] Das Konsortium TopicMaps.org betreibt unter der Adresse http://www.topicmaps.org auch eine Homepage. Auf dieser können weitere Informationen zum Konsortium und zu den Topic Maps gefunden werden.

- *Occurrences*
- *Public Subject Descriptor*
- *Associations*
- *Scopes*
- *Facets*
- *Topic Maps*
- *Bounded Object Sets*

Topics besitzen die Eigenschaften („*characteristics*") Name (*Basename*), zugeordnete Ressourcen (*Occurrences*) und zugeordnete Beziehungen (*Associations*). Eigenschaften von Topics werden in diesem Dokument synonym auch als „Elemente" bezeichnet. Zudem ist es möglich, die Gültigkeit von topics innerhalb eines gewissen Themenbereichs („*scope*") zu beschränken. Topics dienen also dazu, ein bestimmtes **Phänomen der Wirklichkeit** abzubilden, dessen Zweck die Speicherung oder Vermittlung von Wissen ist.

Beziehungen (Assoziationen) zwischen Topics dienen dazu, ein semantisches Netz aufzubauen.

Topic Maps sind untereinander vernetzte *Topics*. Mit anderen Worten: Dokumente, welche der XTM Spezifikation ebenfalls gehorchen.

In der Folge wird auf zwei Aspekte des XTM Modells genauer eingegangen. Abschnitt 4.4.2.1 behandelt wichtige Designziele, die in die Spezifikation des XTM Modells eingeflossen sind. Abschnitt 4.4.2.2 gibt einen Überblick über das komplexe konzeptuelle Modell der XML Topic Maps. In Abschnitt 4.4.2.3 konzentriere ich mich auf die detaillierte Analyse der Konzepte in XTM. Schließlich erfolgt in Abschnitt 4.4.2.4 eine Übersicht, wie der XTM Standard in der Praxis umgesetzt wurde.

4.4.2.1 Designziele der XTM (design goals)

In [XTM, 2001] werden eine Reihe von Designzielen genannt, die bei der Definition des Modells berücksichtigt wurden.

- Das geschaffene Modell muss im gesamten Internet **universell verwendbar** sein. („XTM shall be straightforwardly usable over the Internet.”)
- Eine große **Vielzahl von Anwendungsmöglichkeiten** muss garantiert werden. („XTM shall support a wide variety of applications.")
- **Kompatibilität** mit XML, XLink und dem ISO/IEC 13250 Standard ist sicherzustellen.
- Die Erstellung von Programmen, welche auf Grundlage von XTM erstellte Dokumente verarbeiten, muss einfach sein. Mit anderen Worten, die **Schnittstelle** zur Bearbeitung von XTM Dokumenten muss möglichst einfach und komfortabel sein.
- Die Anzahl **optionaler Features** von XTM muss **minimal** gehalten werden. Im Idealfall sind gar keine optionalen Features vorzusehen.

- XTM Dokumente sollen für den **Menschen lesbar** und möglichst leicht intuitiv verständlich sein. („XTM documents should be human-legible and reasonably clear.”)
- Das **Design**, d.h. der Vorgang zum Entwurf einer Topic Map, muss schnell fertig zu stellen sein. (“The XTM design should be prepared quickly.”)
- Der Entwurf der XTM muss **formal und prägnant** gestaltet sein. („The design of XTM shall be formal and concise.”)
- Das Erstellen von XTM Dokumenten muss so **einfach** wie möglich sein.
- **Knappe Bezeichnungen** für die Tags von XTM sind dagegen nur von untergeordneter Bedeutung. („Terseness of XTM markup is of minimal importance.“)

4.4.2.2 Übersicht über das konzeptionelle Modell der XTM

Mit Hilfe einer einfachen Notation, die auf der UML basiert, werden in diesem Abschnitt die Konzepte schematisch erläutert, die hinter der XML Topic Maps Spezifikation stehen. Dadurch wird versucht, die sehr komplexen Zusammenhänge zwischen den Konzepten leichter fassbar darzustellen. Folgende Konzepte sind für das Verständnis von Topic Maps bedeutend:

- 4.4.2.2.1 Klassenhierarchie (Klassendiagramm)
- 4.4.2.2.2 Klasse-Instanz-Beziehung
- 4.4.2.2.3 Reifizierung von Subjects durch Topics
- 4.4.2.2.4 Referenzierung eines Subjects
- 4.4.2.2.5 Eigenschaften eines Topics
- 4.4.2.2.6 Namenskonventionen innerhalb eines Gültigkeitsbereichs
- 4.4.2.2.7 Verbindungen bzw. Vorkommen von Topics
- 4.4.2.2.8 Assoziationen zwischen Topics
- 4.4.2.2.9 Topic Maps

4.4.2.2.1 Klassenhierarchie (Klassendiagramm)

Abbildung 24 zeigt die Klassen, die im XTM Modell von Bedeutung sind. Um Missverständnisse durch unklare Übersetzungen zu vermeiden, wird die englische Originalbeschreibung des Klassendiagramms angeführt.

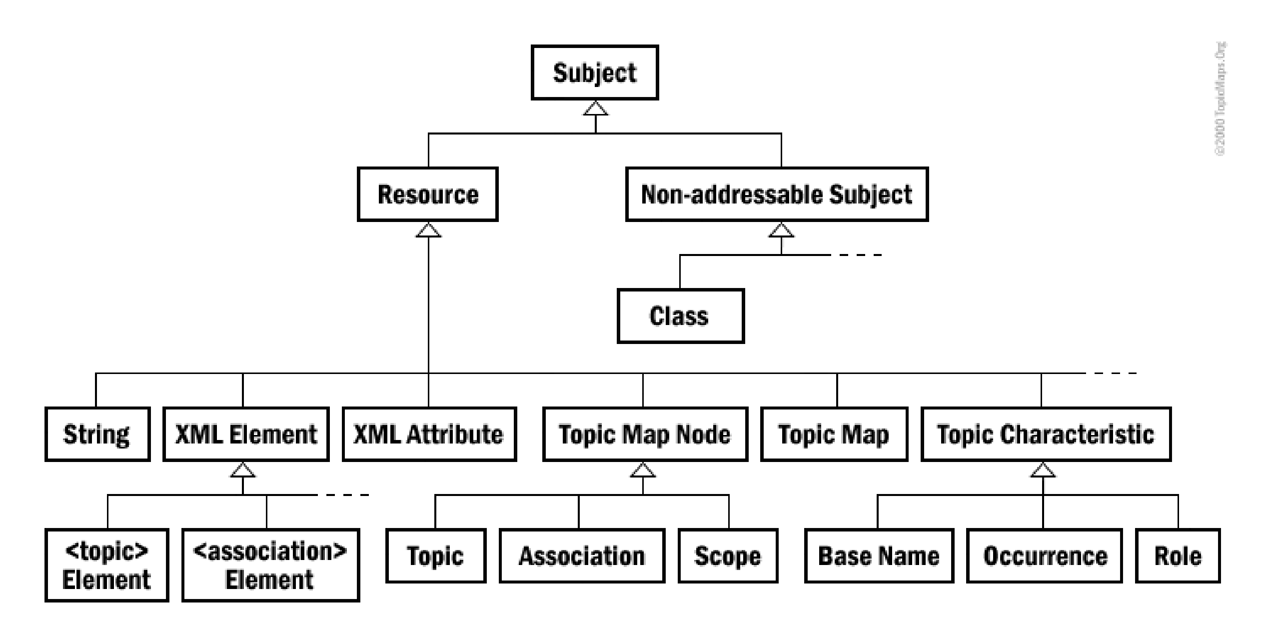


Abbildung 24: Klassendiagramm: XML Topic Maps [XTM, 2001]

"A Subject is anything that can be spoken about or conceived of by a human being. A Resource is a Subject that has identity within the bounds of a computer system. Any other Subject is known as a Non-addressable Subject. There are many types of Non-addressable Subject. A Class is a Non-addressable Subject. Types of Resource include String, XML Element, and XML Attribute, as well as Topic Map, Topic Map Node and Topic Characteristic, and many others. Types of XML Element include `<topic>` Element and `<association>` Element, and many others. There are just three types of Topic Map Node: Topic, Association, and Scope. There are just three types of Topic Characteristic: Base Name, Occurrence, and Role." [XTM, 2001].

4.4.2.2.2 Klasse-Instanz-Beziehung

Ein *Subject* kann die Instanz **einer oder mehrerer** Klassen sein, wie in Abbildung 25 veranschaulicht.

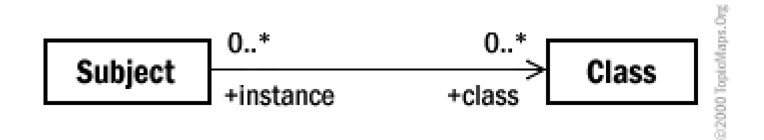


Abbildung 25: Klasse-Instanz-Beziehung: XML Topic Maps [XTM, 2001]

"A Subject may be an instance of zero or more Classes." [XTM, 2001].

4.4.2.2.3 Reifizierung von Subjects durch Topics

Reifizierung[20] („Vergegenständlichung") im Sinne des Topic Map Paradigmas ist der Vorgang der **Abbildung eines Phänomens** aus der wirklichen Welt. Mit anderen Worten ist damit die Bildung (Definition) von *Subjects* und die anschließende Zuordnung eines oder mehrerer Topics auf das *Subject* gemeint. Zudem erlaubt die Reifizierung eine Zuordnung von Eigenschaften zu Topics (vgl. Abschnitt 4.4.2.2.5 Eigenschaften eines Topics, Seite 83). Abbildung 26 veranschaulicht das Konzept der Reifizierung.

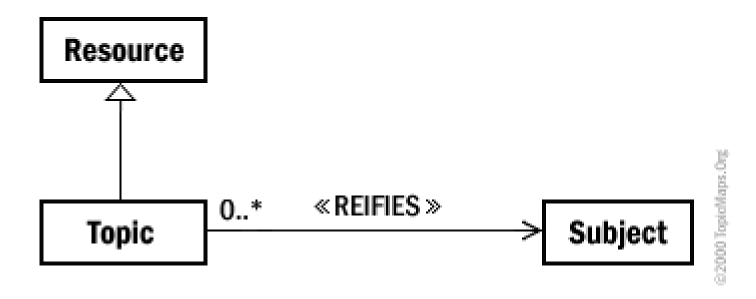


Abbildung 26: Konzept der Reifizierung [XTM, 2001]

"A Topic is a Resource that reifies a Subject. It is the Topic Map system's representation of the Subject. Reification of a Subject allows Topic Characteristics to be assigned to the Topic that reifies it." [XTM, 2001].

4.4.2.2.4 Referenzierung eines Subjects

Subject Indicators sind Bezeichnungen für die (semantische) **Bedeutung**, die ein *Subject* besitzt und die ihm folglich zugeordnet werden kann. Der *Subject Indicator* ist immer eine Ressource; auch ein *Subject* kann, muss aber nicht eine Ressource sein.

Ein Topic kann eine Referenz auf eine Ressource besitzen genau dann, wenn das *Subject* eine Ressource ist. Diese Referenzen existieren völlig unabhängig von den Referenzen auf *Subject Identifiers*. Abbildung 27 zeigt die verschiedenen Möglichkeiten der Referenzierung von *Subjects* durch Topics.

[20] von lat. res: Sache

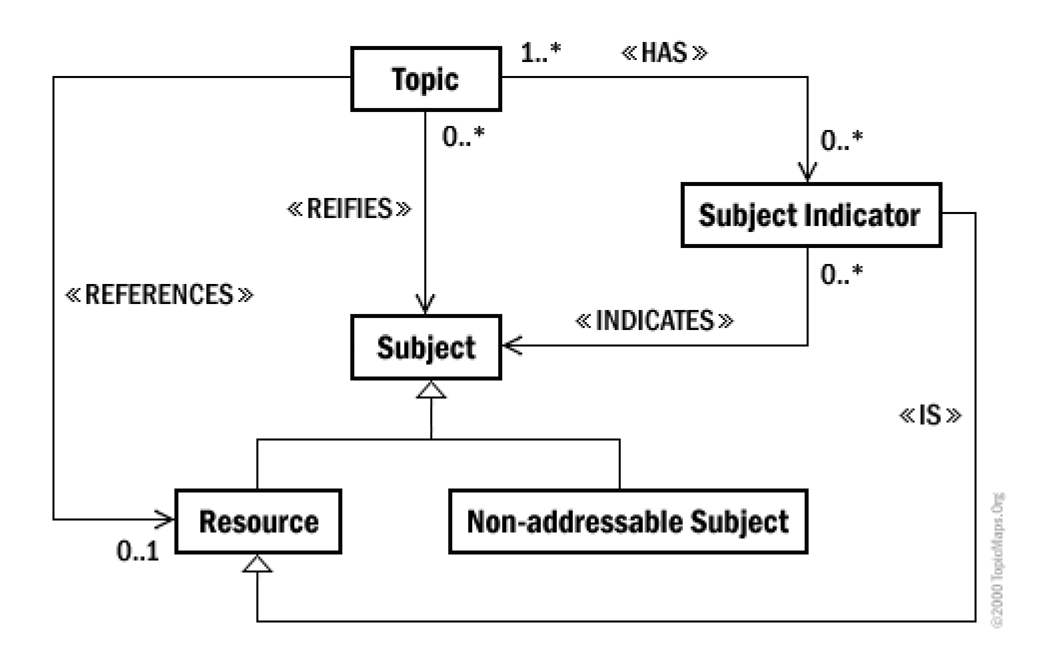


Abbildung 27: Möglichkeiten zur Refenzierung: Topic → Subject [XTM, 2001]

"A Topic can have any number of Subject Indicators. A Subject Indicator is a Resource that indicates what Subject is reified by the Topic. If the Subject is itself a Resource, there can be a direct reference from the Topic to that Resource in addition to any references there may be to Subject Indicators." [XTM, 2001].

4.4.2.2.5 Eigenschaften eines Topics

Eine Eigenschaft eines Topics kann grundsätzlich alles sein, das über ein Topic **behauptet werden kann**. Diese grundsätzlich weit reichende und freie Definition der Eigenschaften wird freilich durch das Konzept weiter eingeschränkt. Ein **Gültigkeitsbereich** (*Scope*) eines Fachgebiets wird durch eine Menge von Topics definiert. Dieser legt fest, welche Eigenschaften einem Topic gültig zugeordnet werden können. Abbildung 28 veranschaulicht dieses Konzept.

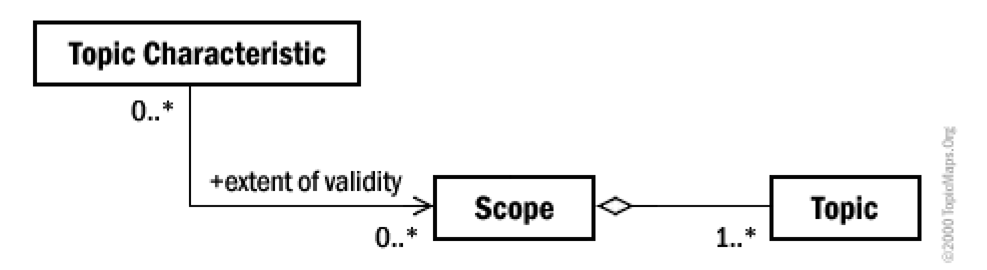


Abbildung 28: Modellierung von Eigenschaften eines Topics [XTM, 2001]

"A Scope is set of Topics that defines the extent of the **validity** of the assignment of a Topic Characteristic to a Topic. If no Scope is specified, the Scope is deemed to be the unconstrained Scope, and the assignment is always valid." [XTM, 2001].

4.4.2.2.6 Namenskonventionen innerhalb eines Gültigkeitsbereichs

Einem Topic wird eine Menge von **Basisnamen** („*Basenames*", in der Quelle [XTM, 2001] als „*base names*" bezeichnet) zugeordnet, welche innerhalb eines bestimmten Fachbereichs („*scope*") Gültigkeit besitzen. *Basenames* sind Zeichenketten und werden für die Definition von Topic-Eigenschaften heran gezogen. Auf diese Weise wird eine Namenskonvention festgelegt, so dass die Modellierung eines **Namensraums** möglich ist. Abbildung 29 verdeutlicht das Konzept.

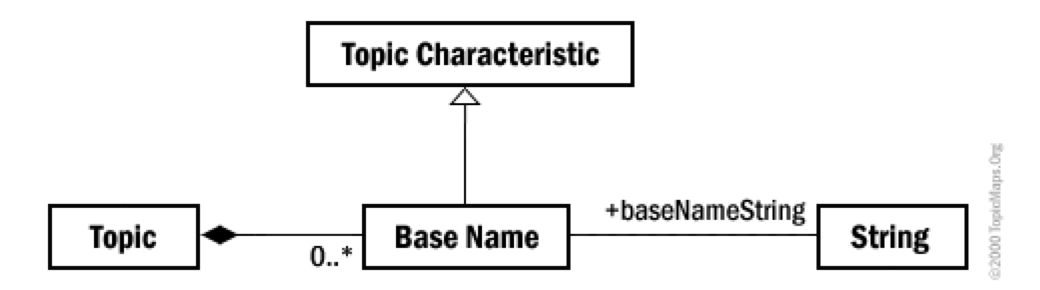


Abbildung 29: Schaffung von Namenskonventionen: XML Topic Maps [XTM, 2001]

"A Base Name is a String that is used to name a Topic within a Scope. Only one Topic may be assigned a particular Base Name within a given Scope. The set of Base Names assigned within a given Scope thus constitutes a **namespace**, and may be used to identify Topics unambiguously." [XTM, 2001].

4.4.2.2.7 Verbindungen bzw. Vorkommen von Topics

Das Konzept der *Occurrences* (Verbindung bzw. Vorkommen) verknüpft bzw. verbindet das mit Hilfe der Topics geschaffene semantische Netz mit physischen Ressourcen. Mit anderen Worten, *Occurrences* sind das **Bindeglied** zwischen einer Menge von **digitalen Ressourcen** und dem darüber gelegten **semantischen Netz**. Semantisches Netz und digitale Ressourcen können voneinander völlig unabhängig existieren. Abbildung 30 veranschaulicht die Modellierung von *Occurrences*.

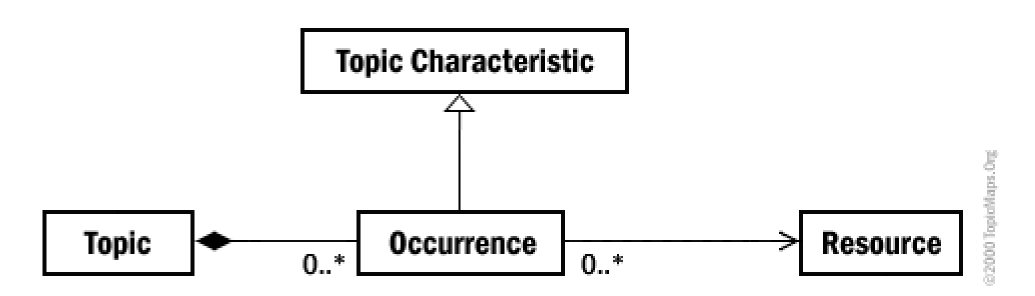


Abbildung 30: Vorkommen von Topics (Occurrence-Konzept) [XTM, 2001]

„An Occurrence designates a Resource that relates to a Topic." [XTM, 2001].

4.4.2.2.8 Assoziationen zwischen Topics

Assoziationen („*Associations*") sind das Konzept, mit dessen Hilfe **semantische Beziehungen** zwischen Topics modelliert werden können. Somit handelt es sich um das jenes Element des Modells, das überhaupt erst das Entstehen eines semantischen Netzes ermöglicht.

Mit Hilfe einer *Association* können Topics untereinander vernetzt werden. Eine Assoziation wird durch eine oder mehrere **Rollen** semantisch definiert. D.h. es gibt spezielle Topics, die dazu dienen, mögliche Rollen abzubilden, welche ein assoziiertes Topic einnehmen kann. In Abbildung 31 wird das Assoziationskonzept veranschaulicht.

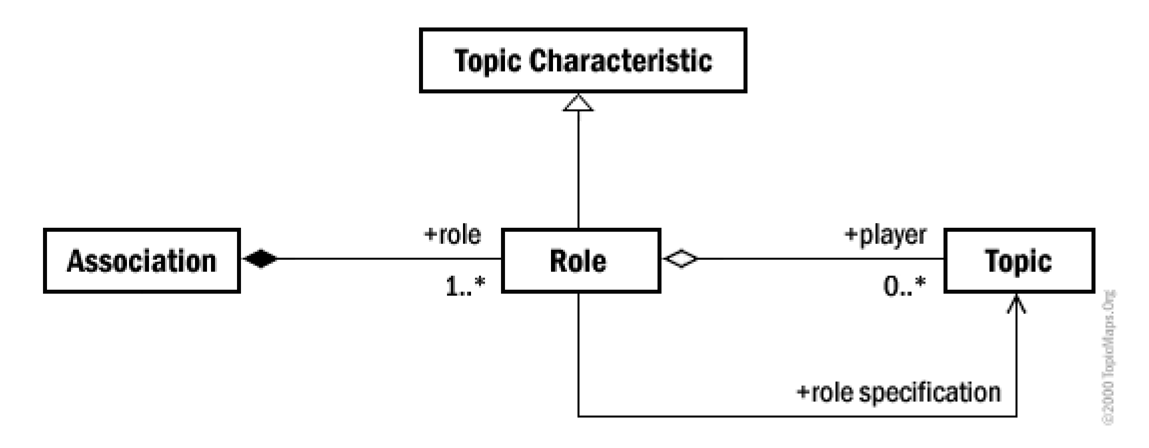


Abbildung 31: Assoziationen zwischen topics (semantische Beziehungen) [XTM, 2001]

"An Association relates Topics to one another. It comprises one or more Roles, each of which corresponds to a Topic that specifies a **type of involvement** that Topics may have in the Association. Each Role is assigned to zero or more Topics that are involved in the Association in the way specified. These Topics are said to be **players** of that Role in the Association.

NOTE: In XTM, it is not permissible for the different Roles in an Association to be governed by different Scopes. The XTM syntax expresses the Scopes on all the Roles of an Association through a single `<scope>` subelement of the `<association>` element." [XTM, 2001].

4.4.2.2.9 Topic Maps

Eine Topic Map besteht aus **beliebig vielen** (meist mehreren) *Topic Map Nodes*. Diesen ist jeweils ein Topic zugeordnet. Topic Maps sind somit immer eine **Teilmenge** aus einem semantischen Netz, welches mit Hilfe von Topics und den Assoziationen zwischen ihnen modelliert wird.

Von einer **konsistenten Topic Map** spricht man, wenn diese ausschließlich *Subjects* enthält, die durch **genau ein** Topic abgebildet sind. Abbildung 32 zeigt, wie Topic Maps mit *Topics* und *Subjects* in Verbindung stehen.

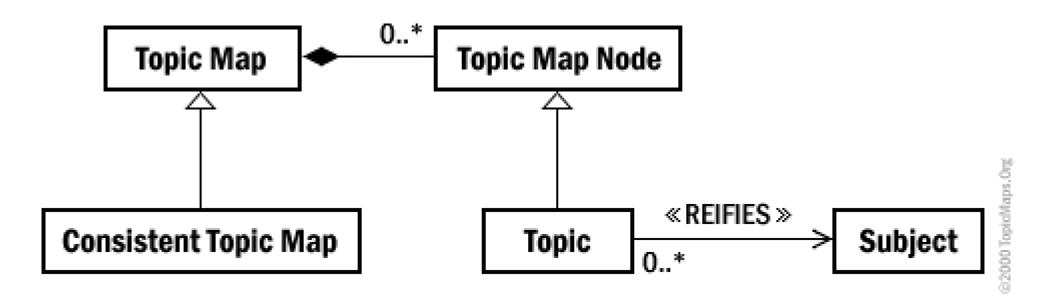


Abbildung 32: Topic Map Konzept [XTM, 2001]

"A **Topic Map** comprises zero or more Topic Map **Nodes** (Topics, Scopes, and Associations). It is possible for more than one Topic in the Topic Map to reify the same Subject. If no Subject is reified by more than one Topic in the Topic Map, then the Topic Map is said to be a **Consistent Topic Map**." [XTM, 2001].

4.4.2.3 Konzepte der XTM (concepts)

Auf Grundlage der in Abschnitt 4.4.2.1 genannten Designziele wurden in [XTM, 2001] die Konzepte definiert, auf denen letzten Endes auch das erstellte XML-Schema basiert. Diese wurden im Abschnitt 4.4.2.2 grundlegend dargestellt. In diesem Abschnitt werden diese Konzepte detailliert erläutert. Der Abschnitt ist in folgende Punkte unterteilt:

- 4.4.2.3.1 Themen („Topics") und Gegenstände („Subjects")
- 4.4.2.3.2 Typen von Themen („Topic Types")
- 4.4.2.3.3 Name eines Topics („Name")
- 4.4.2.3.4 Verbindung bzw. Vorkommen („Occurrences")
- 4.4.2.3.5 Assoziationen („Associations")
- 4.4.2.3.6 Topic Maps

Eine ganz zentrale Rolle nehmen im XTM Modell die Topics ein: mit deren Hilfe wird ein semantisches Netz aufgebaut, in dem neben semantischen Beziehungen praktisch alle Metadaten zu einer Wissensbasis gesammelt werden können.

4.4.2.3.1 Themen („Topics") und Gegenstände („Subjects")

Topics können wohl am Treffendsten als eine Abbildung eines **Phänomens der Wirklichkeit** beschrieben werden. Ein Topic ist „ein elementares Subjekt im Kontext des modellierten Wissens, eine Entität" ([Widhalm, 2002], S. 6).

Die Autoren der XTM Spezifikation wählen folgende Formulierung: „A Topic is a ressource that acts as a proxy for some Subject." [XTM, 2001], S. 17. Wie schon weiter oben erwähnt, umfasst der Begriff in etwa dieselbe Bedeutung wie der Begriff Wisseneinheit. Daher werden die beiden Bezeichnungen in diesem Dokument synonym verwendet.

Das Verhältnis zwischen Wisseneinheit und Phänomen der Wirklichkeit bezeichnen die Autoren als **Reifizierung** (Vergegenständlichung – *„reification"*). Durch den Reifizierungsvorgang wird die Zuweisung von Eigenschaften zu einer Wisseneinheit ermöglicht.

Nachfolgend werden die Konzepte beschrieben, die mit Topics in engem thematischen Zusammenhang stehen.

- **„Subject“**: Der Gegenstand bzw. Themenbereich eines Topics. Unter *Subject* verstehen die Autoren jedes Phänomen der Wirklichkeit, über das Menschen in der Lage sind, zu sprechen oder es zu begreifen. Es gibt zwei Möglichkeiten, wie ein *Subject* aussehen kann. Entweder handelt es sich um ein Dokument, das mit Hilfe eines **URI eindeutig identifizierbar** ist; oder ein *Subject* behandelt ein **abstraktes, nicht adressierbares** Phänomen. Üblicherweise wird einem Topic genau ein Themenbereich zugeordnet; ist diese Eigenschaft erfüllt, spricht man von einem „**konsistenten** Topic“. Besteht eine Topic Map ausschließlich aus konsistenten Topics, so ist auch die Topic Map konsistent. Aufgrund des Reifizierungsprozesses, also des Prozesses der Abbildung von Phänomenen der Wirklichkeit auf Topics, sind die Themenbereiche (*Subjects*) jene Eigenschaft, die das Organisationsprinzip von Topics darstellt. Mit anderen Worten: Topics sind eine abstrakte Vergegenständlichung eines Themenbereichs und existieren abhängig von der Existenz dieses Themenbereichs.
- **„Reification“**: Reifizierung, Vergegenständlichung eines Themenbereichs. Der Begriff meint im engeren Sinne den Vorgang zur Erstellung eines Topics („The act of creating a Topic is called reification.“). Aufgrund der Reifizierung wird das abgebildete Phänomen dem kreierten Topic als *Subject* zugeordnet. Man kann diesen Vorgang daher als die **Abbildung** eines Phänomens der Wirklichkeit in das semantische Netz mit Hilfe der Schaffung eines Topics betrachten. Die Autoren bezeichnen die Reifizierung als jenen Schritt, der es ermöglicht, im Rahmen und mit den Mitteln des Topicmap-Paradigmas über einen Themenbereich aus der Wirklichkeit zu sprechen.
- **„Subject Identity“**: dabei handelt es sich um eine vom Autor (der Topic Map) vergebene, **subjektive Beschreibung** des Gegenstands oder Themenbereichs, von dem das Topic handelt. Am Treffendsten kann man diese Eigenschaft als **Beschlagwortung** übersetzen. *Subject Identities* sind ein Konzept, mit dessen Hilfe innerhalb einer Topic Map herausgefunden werden kann, welche Topics sich mit einem bestimmten *Subject* befassen. *Subject Identities* dienen folglich dazu, den semantischen Inhalt eines Topics zu beschreiben.
- **„Subject Identifier“**: hierbei handelt es sich eine standardisierte Bezeichnung für den Gegenstand oder Themenbereich des Topics. Vorsicht ist geboten: *Subject Identifiers* sind nicht zu verwechseln mit *Subject Identities*, obwohl diese in einem engen thematischen Zusammenhang stehen. Ein *Subject Identifier* ist eine Ressource, die eine abstrakte, unzweideutige Darstellung einer *Subject Identity* darstellt. *Subject Identifiers* sind also **Schlüsselwerte**, die *Subject Identities* identifizieren. Sie werden bei der Erzeugung von Topic Maps verwendet, um zu entscheiden, ob vorgegebene, bereits vorhandene Topics in eine Topic Map aufzunehmen sind oder nicht. Von besonderer Bedeutung sind *Subject Identifiers*, wenn zwei oder mehrere Topic Maps von verschiedenen Verfassern zusammen geführt werden sollen. *Subject Identifiers* vermeiden, dass zwei Topics in einer Topic Map existieren, die das gleiche Phänomen beschreiben, aber unterschiedlich

benannt sind. Ein Beispiel für *Subject Identifiers* sind die von der International Standards Organization ISO standardisierten Codes für die Bezeichnung von Staaten der Erde, wenn sie für *Subject Identifiers* von Topics in einer Topic Map zur Modellierung von Sprachen herangezogen werden.

- **„Topic Characteristics"**: eine Eigenschaft (*Characteristic*) eines Topics ist im weitesten Sinne alles, das man im Rahmen des Topicmap-Paradigmas über das Topic behaupten kann. Bei Characteristics handelt es sich um ein Konzept, das die Modellierung von **Behauptungen über das Topic** mit Hilfe von Referenzen erlaubt. Eigenschaften im Sinn von Characteristics können ein **Name** (siehe Punkt 4.4.2.3.2), eine **Verbindung** („*Occurrence*", siehe Punkt 4.4.2.3.4) oder eine **Rolle** im Kontext einer Assoziation („*Association Role*", siehe Punkt 4.4.2.3.5) sein. Auf die genannten Aspekte wird in den erwähnten Abschnitten noch genauer eingegangen. Die Zuweisung von Eigenschaften muss innerhalb des Gültigkeitsbereichs des Topics („*Scope*") zulässig sein.
- **„Scope"**: der Horizont eines Topics innerhalb eines Fachgebiets. Der Begriff „Horizont" im Sinne der *Scope*-Eigenschaft ist sehr ambivalent. Das Element *Scope* kann verwendet werden, um folgende Zusammenhänge abzubilden: das Ausmaß der **Gültigkeit** der Zuweisung von Eigenschaften (vgl. *Topic Characteristics*); den **Kontext**, in welchem Zuordnungen durch Assoziationen oder Referenzen geschehen können (dieser Horizont kann entweder explizit oder implizit angegeben sein); zuletzt kann auch noch ein **Namensraum** für die Namen von Topics definiert werden. Ein Namensraum für Topic Names wird von den Autoren als *„Topic Naming Constraint"* [XTM, 2001] bezeichnet. Damit ist die Einschränkung gemeint, dass alle Topics, welche sich auf einen bestimmten Themenbereich beziehen, mit dem gleichen Namen („*Basename*" – vgl. den Abschnitt 4.4.2.3.3) benannt werden sollten (Namenskonvention).

4.4.2.3.2 Typen von Themen („Topic Types")

Topics können auch (im Sinne einer Klassenbildung bei objektorientierter Modellierung) kategorisiert werden. Wie im vorangegangenen Abschnitt erwähnt, sind Topics dazu geeignet, alle denkbaren Phänomene der Wirklichkeit abzubilden. Ein Topic ist ein elementares Subjekt im Kontext des modellierten Wissens. Folglich hängt die Bedeutung der Topics vom modellierten Wissen ab. Es ist ratsam, für die Phänomene, also auch für die Topics, eine **Einteilung in Kategorien** zu ermöglichen.

Der XTM Standard erlaubt daher die Definition von *Topic Types*. Ein Vergleich zum Klassenkonzept in der objektorientierten Programmierung ist hier durchaus angebracht; die Zuordnung eines Topics zum *Topic Type* entspricht einer **Instanz-Klasse-Beziehung**.

Zu beachten ist, dass *Topic Types* selbst als Topic modelliert werden müssen. Die Unterscheidung, welche Topics als Typen verwendet werden können, liegt beim Verfasser der Topic Map bzw. wird durch die Menge der Beziehungen in einer Topic Map implizit ersichtlich.

4.4.2.3.3 Name eines Topics („Name“)

Jedes Topic kann **keinen bis mehrere** Namen besitzen, wobei jeder der vorhandenen Namen lediglich innerhalb eines gewissen Fachbereichs („*Scope*“) Gültigkeit besitzt. Namen können in verschiedenen **Erscheinungsformen** auftreten: als formelle Namen, als symbolische Namen, als Spitznamen, usw.

Namen für Topics zu vergeben ist nicht verpflichtend, aber empfehlenswert. Dadurch wird insbesondere die **Wartbarkeit** des geschaffenen semantischen Netzes verbessert. Jeder Name wird durch drei Merkmale eindeutig identifiziert:

- **„Basename“**: dieser stellt den eigentlichen Namen an sich dar. Der Basename ist eine Zeichenkette, die gemäß der **Namenskonvention** (vgl. Punkt 4.4.2.3.1, Unterpunkt „*Scope*“) ausgewählt werden sollten. Die Verletzung der Namenskonvention beeinträchtigt die Wiederverwendbarkeit der Topics im semantischen Netz.
- **„Variant Name“**: Bezeichnung der Art dieses Namens. Hier handelt es sich, ähnlich wie bei *Topic Types* (vgl. Punkt 4.4.2.3.2), um eine Definition von Klassen für Namen. Je nach Fachgebiet („*Scope*“) sind verschiedene Arten (Klassen, Varianten) von Namen sinnvoll. Eine eigene Definition **benutzerdefinierter** Namensvarianten wird unterstützt.
- **„Parameters“**: Parameter sind zusätzliche Angaben zum *Variant Name*, in denen Angaben über deren optimalen Geltungsbereich gespeichert werden können. Diese Angaben werden in Form einer Menge von Referenzen auf Topics gespeichert. In den Parametern kann z.B. gespeichert werden, dass ein bestimmtes Topic für die Einbindung in deutschsprachigen Dokumenten geeignet ist oder dass es zum Fachbereich „Wirtschaftsinformatik“ gehört.

Der Topic Map Standard erlaubt drei verschiedene Varianten von Namen: *Base Names*, *Display Names* und *Sort Names*. Die genannten Varianten sind leicht unterschiedlich zum Namenskonzept in XTM und werden daher in der Folge kurz erläutert (vgl. [Widhalm, 2002]).

- *Base Names*: dies ist der „eigentliche Name“ eines Topics. In verschiedenen Gültigkeitsbereichen (*Scopes*) kann ein Topic mehrere solcher Namen aufweisen, wobei es jedoch mindestens einen Namen besitzen muss.
- *Display Name*: dies ist eine Zeichenkette, die für die Anzeige oder Darstellung eines Topics herangezogen wird. Die Angabe eines oder mehrerer Anzeigenamen ist optional. Gibt es keinen *Display Name*, übernimmt der *Base Name* dessen Funktion.
- *Sort Name*: hierbei handelt es sich um eine Zeichenkette, die für Sortierzwecke (z.B. in sortierten Listen) verwendet wird. Wie beim *Display Name* ist auch die Angabe eines Sortiernamens optional. Ist kein solcher vorhanden, übernimmt der *Base Name* dessen Funktion.

4.4.2.3.4 Verbindung bzw. Vorkommen („Occurrences“)

Verbindungen, oder vielleicht treffender „Vorkommen“, von Topics bezeichnen jede Art von **Ressource**, die für ein bestimmtes Topic **relevant** sind, wobei sich diese Ressour-

cen selbst nicht innerhalb des semantischen Netzes befinden. Mit anderen Worten bezeichnen die *Occurrences* eines Topics jene Phänomene, zu denen eine semantische Zuordnung des Topics zutreffend ist. Meist handelt es sich bei diesen Phänomenen um **elektronische Dokumente** jeglicher Art, die durch URIs identifiziert werden können. Zum Beispiel wäre ein Dokument, in dem das Stück Hamlet aufgezeichnet ist, dem Topic „Shakespeare" als *Occurrence* sinnvoll zuordenbar.

Verbindungen besitzen zwei zentrale Eigenschaften: die Rolle der Verbindung („*Occurrence Role*") sowie die Art der Verbindung („*Occurrence Type*"). Es ist wichtig, diese beiden Punkte näher zu erläutern.

- **„Occurrence Role"**: die Rolle der Verbindung ist eine mehr oder weniger beliebige Benennung der Beziehung; sie kann als **Gedächtnisstütze** („*mnemonic*") betrachtet werden. Sie gibt nicht zwingenderweise Aufschluss über die semantische Bedeutung der Beziehung.
- **„Occurrence Type"**: die Angabe über die Art der Verbindung enthält wichtige semantische Information. Sie präzisiert die Art der **Relevanz der Verbindung** für das referenzierte Topic. Vielleicht ist dies schwer erkennbar, doch es handelt sich auch hier um eine Klassenbildung für *Occurrence*-Beziehungen. Beispiele für Klassen (Arten) der Verbindung sind „Monografie", „Artikel", „Illustrationsbeispiel", „Formel", usw.

Eine präzise Modellierung von Verbindungs-Beziehungen ist von besonderer Bedeutung für den Wert und die Wiederverwendungsfähigkeit von Wissensbasen, die mit Hilfe semantischer Netze (wie z.B. den Topic Maps) modelliert sind.

4.4.2.3.5 Assoziationen („Associations")

Bis jetzt wurde noch kein Konzept besprochen, durch das Beziehungen, d.h. semantische (inhaltliche) Verknüpfungen, von Topics untereinander modelliert werden können. Die Definition von Assoziationen schließt diese Lücke.

Mit Hilfe von Assoziationen können inhaltliche Beziehungen zwischen zwei oder mehreren Topics geschaffen werden. Assoziationen zwischen Topics sind der **entscheidende Schritt von Information zu Wissen**. Assoziationen erlauben es, Aussagen wie „Rom liegt in Italien" in das semantische Netz abzubilden, indem für Rom und für Italien je ein Topic angelegt wird. Die semantische Beziehung zwischen den beiden Topics wird durch eine Assoziation gewonnen, wobei ein Weg gefunden werden muss, auch die Bedeutung derselben („liegt in") beim Modellierungsprozess nicht zu verlieren.

Das genannte Beispiel zeigt anschaulich, welche Aspekte einer Assoziation gespeichert werden müssen. Zum einen müssen die **beteiligten Topics** nachvollziehbar sein; zum anderen ist eine Speicherung der **Art der Assoziation** erforderlich. Die Notwendigkeit, weitere Information zu speichern, ist nicht auf den ersten Blick ersichtlich. Eine Assoziation ergibt immer nur in einer Richtung einen Sinn [Pepper, 2002]. Um auch die **Richtung der Assoziation** angeben zu können, muss für Topics eine Rolle gespeichert werden, die sie im Rahmen der Assoziation einnehmen. So kann die Fehlinterpretation „Italien liegt in Rom" ausgeschlossen werden.

Die eben erwähnten Fakten werden in den folgenden Objekten physisch gespeichert. Die wichtigste Information zu jedem Objekt wird nochmals zusammen gefasst:

- **„Member"**: speichert alle an der Assoziation beteiligten Topics.
- **„Role"**: die Rolle, die ein Topic in einer Assoziation einnimmt. Auf diese Weise werden, wie erwähnt, gerichtete Assoziationen modelliert. Die Benennungen von Rollen sollten sich ebenfalls nach der Namenskonvention (vgl. Punkt 4.4.2.3.1, Unterpunkt *„Scope"*) richten.
- **„Association Type"**: ein Typ von Assoziation. Wiederum handelt es sich um eine Klassenbildung, hier um die Bildung von Klassen für Assoziationen.

4.4.2.3.6 Topic Maps

Als letzter Schritt zu einem vollständigen Modell fehlt lediglich die Möglichkeit, **ganze Dokumente** zu erstellen und zu speichern. Dies geschieht mit Hilfe von Topic Maps. Der XTM Spezifikation [XTM, 2001] kann eine XML Document Type Definition (dtd) entnommen werden, in der die Modellierung von Topic Maps im XML-Format definiert ist. Topic Maps besitzen folgende Charakteristika:

- Topic Maps sind eine **Sammlung** von Topics, Assoziationen und Namensräumen (*Topics*, *Associations*, *Scopes*).
- Sie können in zweierlei Form existieren: entweder als **serialisiertes** und somit austauschbares XML-Dokument, oder lediglich im **Arbeitsspeicher** als applikationsspezifische Repräsentation.
- Eine Topic Map ist **konsistent**, wenn zwischen Topics und *Subjects* eine 1:1-Zuordnungsrelation besteht. D.h. wenn jedem Topic nur genau ein *Subject* zugeordnet wird und umgekehrt.
- Von einem **Topicmap-Dokument** (*„Topicmap Document"*) spricht man, wenn ein Dokument eine oder mehrere Topic Map(s) enthält, welche wohlgeformt sind, also der ISO/IEC Spezifikation entsprechen. Topicmap-Dokumente können zum Zweck der permanenten Speicherung oder des Austauschs serialisiert werden.
- Ein **XTM Dokument** ist eine Topic Map, die nicht nur zur ISO/IEC Spezifikation, sondern auch zur XTM Spezifikation konform ist.

4.4.2.4 Umsetzung der Topic Map Spezifikation

Wie bereits am Beginn des Abschnitts über Topic Maps erwähnt, wurde das Topic-Map-Paradigma von der International Standards Organization (ISO) in einem Standard festgeschrieben (vgl. [ISO 13250/1999] und [ISO 13250/2000]). Der Standard wurde so weit wie möglich unabhängig von einer Implementierungssprache definiert. Zu diesem Zweck wurde die allgemeine Auszeichnungssprache SGML verwendet [ISO 13250/1999].

Mittlerweile gibt es **mehrere Implementierungen** des XML Topic Maps Standard. Diese unterscheiden sich nur geringfügig, da sie auf der Grundlage des ISO/IEC Standards entworfen wurden. Die Unterschiede zwischen den Implementierungen sind dennoch vorhanden und werden hier erläutert.

An Implementierungen für den XTM Standard sind folgende Konzepte verfügbar:

- XML Topic Maps Document Type Definition (DTD) des Konsortiums Topicmaps.org (`xtm1.dtd`): die vom offiziellen Topic-Map-Konsortium stammende DTD stellt die **erste Umsetzung** des ISO/IEC Topic Map Standards dar. Nach Eigendefinition handelt es sich dabei um ein Format, das den Austausch von Topic Maps auch über kulturelle Grenzen hinweg ermöglicht [XTM dtd, 2001]. Alle im ISO/IEC Topic Map Standard vorgesehenen Elemente werden definiert.
- Ontopia DTD Topic Map Interchange Format von Ontopia AS[21] (`tm_strict.dtd`): hierbei handelt es sich um ein Format des Unternehmens Ontopia AS für den Austausch von Topic Maps, bei dem die meisten der im Standard vorgesehenen **Freiheiten eingeschränkt** sind. Folgende Einschränkungen sind im Vergleich zum Standard implementiert (vgl. [TM Strict, 2000]):
    - Jedes der Objekte Topic, Occurrence, Association und Facet **muss** mit Hilfe der type-of Beziehung klassifiziert werden.
    - Implizit definierte Topics (z.B. Topics, die in einer Topic Map durch Verweis auf eine andere Topic Map „importiert" werden) sind nicht erlaubt. Jedes Topic muss explizit definiert werden.

  Durch die strenge und eingeschränkte Definition der Elemente kann die DTD von Ontopia verwendet werden, um Topic Maps zu validieren (vgl. [TM Strict, 2000]), d.h. zu überprüfen, ob sie im Sinn von XML wohlgeformt sind.
- Ontopia Linear Topic Map (LTM) Notation: die LTM-Notation ist ein **proprietäres Format** des Unternehmens Ontopia AS[21]. LTM stellt eine gegenüber des XML-Codes bedeutend **kürzere Notationssprache** für Topic Maps zur Verfügung. Mit Hilfe eines Interpreters („LTM Processor") kann eine mit LTM beschriebene Topic Map in eine XML Topic Map übersetzt werden [Garshol, 2002]. Autoren von Topic Maps können durch die Verwendung der LTM Notation den Schreibaufwand für die Erstellung einer Topic Map reduzieren.

4.5 Referenzprojekte und Rahmenmodelle

In diesem Abschnitt werden Projekte aus Wissenschaft und Praxis untersucht, die sich mit ähnlichen Problemstellungen wie das Projekt „Entwicklung eines Contentpools für Scholion WB+" beschäftigen. Dazu gehören auch Projekte, die ein Rahmenmodell (*Framework*) für die Zerlegung von Dokumenten, die Speicherung, Darstellung und Verwendung von zerlegten Dokumenten bereitstellen. Gemäß der Gliederung der Literaturanalyse (siehe Abschnitt 4.1.2) werden im Folgenden behandelt:

[21] Siehe http://www.ontopia.net

4.5.1 Trial Solution Projekt

Trial Solution ist ein Akronym und steht für Tools for Reusable, Integrated, Adaptable Learning – Systems/Standards for Open Learning Using Tested, Interoperable Objects and Networking. Das Projekt wird von der Europäischen Union im Rahmen des Forschungsprogramms **Information Society Technologies Programme** (IST) finanziert. Das IST Forschungsprogramm ist einer der Schwerpunkte der technologischen Forschung und Entwicklung innerhalb des Fünften Rahmenprogramms der Europäischen Gemeinschaft im Bereich der Forschung, Technologischen Entwicklung und Demonstration (FTE – im englischen Original „European Community for Research and Technological Development" RTD – vgl. [RTD, 2002]).

Ziel des Trial Solution Projekts ist die Entwicklung von Methoden, Konzepten und Algorithmen zur **Zerlegung** von Mathematik-Lehrbüchern und die **Speicherung** der generierten Wissenseinheiten in einer Wissensbasis. Wesentlicher Eckpfeiler des Projekts ist die Evaluierung und Weiterentwicklung der Slicing Book Technology (siehe Abschnitt 4.2.1 ab Seite 43), so dass in Zukunft nicht nur ein einziges Buch, sondern mehrere Bücher in einer Wissensbasis abgespeichert werden können.

Die Laufzeit des Projekts betrug insgesamt drei Jahre. Es wurde im Februar 2000 gestartet und lief bis 30. April 2003 (vgl. [Trial, 2003]).

Im Folgenden werden Ziele des Projekts (Abschnitt 4.5.1.1), die wichtigsten Ergebnisse (Abschnitt 4.5.1.2), sowie die Architektur des geschaffenen Softwaresystems (Abschnitt 4.5.1.3) vorgestellt.

4.5.1.1 Ziele von Trial Solution

Wichtigstes und oberstes Ziel von Trial Solution war es, **neue Wege** sowohl des individuellen als auch des kooperativen Lernens durch die Entwicklung eines mächtigen Konzepts zur Erstellung und Distribution von Dokumenten zu erschließen. Zu diesem Zweck sollten die Anwendungsmöglichkeiten der Slicing Book Technology erprobt und evaluiert werden. Darüber hinaus setzten sich die Projektpartner folgende Ziele [Trial, 2003]:

- **Skalierbarkeit** des Konzeptes: Die entwickelten Methoden und Werkzeuge müssen sowohl für den Einsatz in einer Kleingruppe als auch für die Verwendung bei großen Zuhörerschaften geeignet sein.
- Ergebnis der Arbeiten muss ein *Repository* (eine Wissensbasis) sein, das aus kleinen, **wieder verwendbaren Wissenseinheiten** (Learning Objects) besteht. Die Wissensbasis hat zu bestehen einerseits aus einer Datenbasis mit Wissensmaterialien, und andererseits aus Metadaten zum Inhalt, zu didaktischen Gesichtspunkten und zur Kopplung mit anderen Wissensbasen.
- Ausgehend von der Wissensbasis muss die Erstellung **individueller Dokumente** durch das Zusammenfügen von Wissenseinheiten möglich sein.
- Im Verlauf der Forschungsarbeiten ist das **Potential** aufzudecken und auszuschöpfen, das in der Verwendung von individualisierten Lernmaterialen schlum-

mert.[22] Es ist ein **Nutzen** zu erwarten beim Einsatz der Lernmaterialien im Unterricht, eine Vereinfachung der Verwendung und der Wartung der Wissensbasis – das alles verbunden mit geringeren laufenden Kosten, höherer Servicequalität und Skalierbarkeit, als es bisher mit „traditionellem Lernmaterial" erreichbar war. Es ist zu evaluieren, wie groß dieser Nutzen ist.

- Bei der Entwicklung von Tools ist zu berücksichtigen, dass es bereits qualitativ hochwertige Formate zur Publikation von Dokumenten gibt. Um die Kosten des Prozesses der Erstellung der Wissensbasis so gering wie möglich zu halten, müssen diese **Dokumentenstandards** berücksichtigt werden. Beispiele für diese Standards sind LaTeX, Rich Text Format, usw.
- Das geschaffene Softwaresystem muss **offen** sein. Es ist eine strenge Trennung der einzelnen Komponenten in **Module** vorzunehmen.
- Die Basis der zu implementierenden Software bilden existierende anerkannte und weitverbreitete **Standards** (z.B. Hypertext-Konzept) sowie Standards, deren Verbreitung absehbar ist (z.B. XML, Learning Object Model [Downes, 2000] – vgl. Abschnitt 4.3.1, Seite 53 ff.).
- Für noch nicht oder nicht vollständig definierte oder ausgereifte Standards ist eine **Vorreiterrolle** bei deren Gestaltung anzustreben.
- Ergebnisse der entwickelten Tools und der geschaffenen Wissensbasis müssen auch in **anderen Anwendungsgebieten** verwendbar sein (reusability), um die hohen Kosten zu ihrer Erstellung zu rechtfertigen.
- Zum Zweck einer möglichst umfassenden Evaluierung der Vorteile und des Nutzens der geschaffenen Methoden und Werkzeuge sind die **Erfahrungen** von Lehrenden, Lernenden, Autoren und Verlagen zu berücksichtigen und zu veröffentlichen.

4.5.1.2 Ergebnisse von Trial Solution

Das Trial Solution Projekt leistete umfassende Beiträge zum weiter oben erwähnten Fünften Rahmenprogramm der FTE, welche in [RTD, 2002] und [Trial, 2003] im Detail nachgelesen werden können. Die Ergebnisse des Projekts lauten wie folgt [TS final, 2003]:

- Die prototypischen Anwendungen des Trial Solution Projekts, die für das Fachgebiet der Mathematik optimiert wurden (vgl. [Trial, 2003], [Dahn, 2001d]), lieferten wertvolle **algorithmische Erkenntnisse**, die leicht zur Anwendung auf anderen Fachgebieten adaptiert werden können. Dies gilt insbesondere für die Methoden und Werkzeuge zur Zertümmerung, Bearbeitung, Datenabfrage (d.h. die Erstellung von Dokumenten) und Distribution.

[22] Anmerkung: bereits im Rahmen des Projekts Slicing Book Technology wurde die Hypothese aufgestellt, dass der Einsatz von individualisierten Lehr- und Lernmaterialien den Lernprozess verbessert, ohne dass diese nachgewiesen wurde. Vgl. dazu Abschnitt 4.2.1, Seite 43 ff.

- Eine umfangreiche **Wissensbasis** wurde geschaffen, deren Metadaten und auch Wissensdaten über das World Wide Web genutzt werden können. Insgesamt wurden rund 5.000 Seiten aus 19 verschiedenen Dokumenten zertümmert und in der Wissensbasis abgespeichert [TS final, 2003].
- Es wurde ein **XML-Schema für Metadaten** auf der Basis von Standards geschaffen und an die Bedürfnisse des Projekts angepasst. Dabei wurde versucht, die (Weiter-) Entwicklung von Standards für die Beschreibung von Wisseneinheiten zu beeinflussen und voranzutreiben.
- Die Schnittstelle zur Suche in der Wissensbasis arbeitet auf Basis **neuer Indizierungstechniken**. Der Suchvorgang läuft in einer wissensbasierten Inferenzkomponente ab (auf Basis der Metadaten, semantischen Verknüpfungen und eines Benutzermodells). Damit ist das System in der Lage, die persönlichen Präferenzen und das Vorwissen eines Benutzers miteinzubeziehen.
- Es wurde ein **Werkzeugkasten** („tool set") geschaffen, mit dessen Hilfe Dokumente personalisiert und aktiv bearbeitet werden können. Der Werkzeugkasten besteht aus folgenden Teilen [TS final, 2003]:
    - Splitter
    - Keyphrase Assignment Tool
    - Reengineering Tool
    - Thesaurus Tool
    - Delivery Tool
    - Knowledge Management System

  Hinweis: Detaillierte Information zu den Werkzeugen des Trial Solution Werkzeugkastens folgt im Abschnitt 4.5.1.3 weiter unten.
- Als Ergebnis der Beschlagwortung und der Erstellung einer Wissensbasis ist zudem ein umfangreicher **Thesaurus** mit rund 16.000 Einträgen aus dem Fachgebiet der **Mathematik** entstanden. Der Thesaurus wurde ausschließlich von Experten entwickelt und wuchs mit Fortdauer des Projekts beständig an (vgl. [TS final, 2003], S. 9).

4.5.1.3 Systemarchitektur

Im Projekt Trial Solution wurde ein **Arbeitsablauf** (*Workflow*) entworfen, der zur automatisierten Zertümmerung von elektronischen Dokumenten und Zusammensetzung neuer Dokumente dient (vgl. Abbildung 33). Dieser Arbeitsablauf bildet zugleich die **Grundlage für die Architektur** von Werkzeugen. Folgende vier Schritte werden genannt (vgl. [Sander, 2001] und [TS final, 2003]):

- Zerlegung elektronischer Dokumente in semantische Einheiten und Aufbereitung der semantischen Einheiten mit Metadaten;
- Versehen der semantischen Einheiten mit Schlagwörtern (*key phrases*);
- Manuelle Bearbeitung der Zerlegung und der Metadaten;

- Generierung neuer personalisierter Dokumente aus den zerlegten Dokumenten.

Jeder der genannten Schritte wird mit einem im Projekt entwickelten Werkzeug unterstützt. Zusätzlich wurden noch ein **Thesaurus** zur Unterstützung der Beschlagwortung sowie eine **Wissensverarbeitungskomponente** (Inferenzmechanismus) zur Unterstützung der Generierung neuer Dokumente implementiert. Somit erhält man sechs Werkzeuge, die im Rahmen des Trial Solution Projekts prototypisch implementiert wurden.

Abbildung 33: Module und Workflow – Trial Solution (Quelle: [Dahn, 2001b], S. 9)

Folgende sechs System-Module (Abbildung 33) bzw. Werkzeuge wurden für Trial Solution prototypisch implementiert (vgl. [TS final, 2003] und [TS Results, 2003]):

- Das Zerlegungstool („*Splitter*"), das zur Zerlegung elektronischer Dokumente in semantische Einheiten dient.
- Ein Werkzeug zur automatischen Zuweisung von Schlagwörtern („*Keyphrase Assignment Tool*").
- Eine webbasierte Schnittstelle für das Steuern der Zerlegung von Dokumenten sowie zur Erstellung und Bearbeitung von Metadaten und einer Beschlagwortung („*Reengineering Tool*").
- Das Werkzeug zur Bearbeitung von mehrsprachigen Thesauri („*Thesaurus Tool*" oder auch „*Trial Solution Metadata Server*").

- Das webbasierte Werkzeug zur Erstellung und Weitergabe von neuen personalisierten Dokumenten aus dem Bestand der semantischen Einheiten in der Wissensbasis („*Delivery Tool*").
- Eine Komponente zur Wissensverarbeitung, in der auf Basis von Metadaten, eines Benutzermodells und generellen Regeln über Lernszenarien Schlussfolgerungen über benötigte semantische Einheiten in einem zu erstellenden Dokument gezogen werden („*Knowledge Management System*").

Alle genannten Werkzeuge wurden für Inhalte auf dem **Fachgebiet der Mathematik** optimiert. Als Resultat wurde aber eine **offene Architektur** auf Basis von Standards geschaffen, so dass diese mit wenig Aufwand auch auf andere Fachgebiete anwendbar ist.

Die Werkzeuge des Trial Solution Projekts werden auf den kommenden Seiten detailliert beschrieben.

4.5.1.3.1 Splitter

Der Splitter baut in großem Umfang auf dem im **Slicing Book Technology** Projekt (vgl. Abschnitt 4.2.1 ab Seite 43) entwickelten **Zerlegungstool** auf. Das Programm wurde unter Regie der Slicing Infotech GmbH[23] entwickelt. Seine Funktionalität kann wie folgt beschrieben werden [TS final, 2003]:

- Automatische **Zerlegung** des in elektronischer Form vorliegenden Quelldokuments;
- Das Erzeugen von wiederverwendbaren „Häppchen" (slices) eines Dokuments, also die Schaffung von **Wissenseinheiten**;
- Erlaubt die Einstellung von **Konfigurationsparametern**, die dem Algorithmus ermöglichen und erleichtern, die Grenzen zwischen Wissenseinheiten zu erkennen. Die Konfiguration der Parameter wirkt sich vor allem auf die Granularität der geschaffenen Wissenseinheiten aus, welche zwischen einer einzelnen Zeile und einem ganzen Kapitel liegen kann.
- **Metadaten** werden soweit wie möglich **automatisch erkannt** und den Wissenseinheiten zugewiesen. Insbesondere Information über die Struktur des Dokuments kann erkannt werden. Schwierigkeiten bestehen jedoch darin, alle anderen Arten von Metadaten zu erkennen, da es bis jetzt noch nicht gelungen ist, einen Algorithmus mit ausreichendem Wissen zur Erkennung von semantischen Zusammenhängen zu bauen.

Fertig zerlegte Dokumente werden vom Splitter im IMS Content Packaging Format gespeichert (vgl. dazu Abschnitt 4.3.3.2, Seite 63 ff.) und in einer Wissensbasis abgelegt [TS final, 2003].

[23] Siehe http://www.slicing-infotech.de/

4.5.1.3.2 Automated Key Phrase Assignment Tool

Das Automated Key Phrase Assignment Tool ist dafür verantwortlich, dass zusätzlich zu der vom Splitter generierten Strukturinformation weitere Metadaten **automatisch generiert** werden. Auf die Schwierigkeiten dabei wurde schon im Punkt 4.5.1.3.1 hingewiesen. Für die Entwicklung des Metadata Tools war das Centrum voor Wiskunde en Informatica in Amsterdam[24] verantwortlich. Die Funktionalität kann folgendermaßen zusammengefasst werden [TS final, 2003]:

- Das Tool erhält als Input die **vom Splitter erzeugten** Wissenseinheiten.
- Eine **Beschlagwortung** des Dokuments wird auf Basis eines eventuell bereits im Dokument vorhandenen Index durchgeführt.
- Das Tool führt grundlegende **Konsistenzprüfungen** der zugewiesenen Metadaten durch. Zu diesem Zweck stehen dem Programm verschiedene Eingabequellen zur Verfügung: eine Tabelle von Standard-Schlagwörtern eines bestimmten Fachgebiets, so genannte „identification clouds“ sowie die Thesauri des Metadata Servers (vgl. den folgenden Punkt). Aus der Quelle ging leider nicht eindeutig hervor, was unter identification clouds zu verstehen ist.

4.5.1.3.3 Metadata Server

Der Metadata Server dient vorwiegend als eine **Datenbank für Schlagwörter**. Das Programm wurde vom Fachinformationszentrum in Karlsruhe[25] entwickelt. Folgende Funktionalität stellt es bereit [TS final, 2003]:

- Ein **Thesaurus** wird angeboten, der dem Key Phrase Assignment Tool (vgl. Punkt 4.5.1.3.2) Synonyme für Fachbegriffe zu einem bestimmten Fachgebiet zur Verfügung stellt.
- Der Metadata Server ist ein **zentraler Pool** für Fachbegriffe zu einem oder mehreren Fachgebieten.
- Wichtig ist besonders, dass Anfragen unterstützt werden, die nicht Synonyme eines Begriffs, sondern in anderer Weise **relevante Begriffe** ermitteln. Beispiele dafür sind „Siehe-auch“ Relationen, spezifischere oder allgemeinere Begriffe.
- Die Erweiterung der Funktionalität auf die Unterstützung von **Mehrsprachigkeit** oder mehrerer Fachgebiete ist leicht durchführbar. Somit ist der Metadata Server ein universell einsetzbares Werkzeug.

4.5.1.3.4 Authoring Tool

Das Bearbeitungstool oder Authoring Tool ist der wichtigste Bestandteil der Architektur des Trial Solution Softwaresystems. Es wird in Zusammenarbeit der Universität Koblenz-

[24] Siehe http://www.cwi.nl/

[25] Siehe http://www.fiz-karlsruhe.de/home.html

Landau[26] und der Université de Nice[27] entwickelt. Das Tool besteht aus einer Client-Komponente und einer Server-Komponente. Die Funktionalität umfasst folgende Punkte [TS final, 2003]:

- Die **Zerlegung** des Dokuments kann manuell nachbearbeitet werden. Mit der Client-Komponente können sowohl zu große Wissenseinheiten aufgespaltet als auch zu kleine Wissenseinheiten zusammengefügt werden. Darüber hinaus sind der nochmalige Aufruf des Splitters und die **Neukonfiguration** der Parameter des Splitters möglich. Das ist besonders für solche Fälle wichtig, in denen der automatisch generierte Output kaum brauchbar ist und eine besonders intensive manuelle Nachbearbeitung erforderlich wäre.
- Vom Metadata Tool und Metadata Server **zugewiesene Metadaten** können manuell bearbeitet, korrigiert und ergänzt werden. Besonders wichtig ist die Erstellung von **Referenzen zwischen** den Wissenseinheiten, also die Herstellung des semantischen Zusammenhangs der Wissensbasis.
- Die Client-Komponente übernimmt für die genannten Aufgaben die **Präsentation** der Zwischenergebnisse an den Benutzer. Sie erlaubt die komfortable und ergonomische Bedienung des Werkzeugs über eine grafische Benutzungsschnittstelle (Graphical User Interface – GUI). Bei der Präsentation ist darauf zu achten, dass von Details der programminternen Datenhaltung **ausreichend abstrahiert** wird.
- Bei der Server-Komponente erfolgt die Abarbeitung der vom Benutzer in Auftrag gegebenen Aktionen. Die **eigentliche Manipulation** des Dokuments wird hier ausgeführt. Es findet eine ständige Kommunikation mit der Client-Komponente statt, so dass der Benutzer in Echtzeit die Ergebnisse seiner Bearbeitungen präsentiert bekommt.
- Die Kommunikation innerhalb des Tools erfolgt vom Client zum Server mit Hilfe einer GET Message des HTML Protokolls. Die Antworten vom Server sind in einem **programmspezifischen Protokoll** verfasst, sind also spezifisch für dieses Werkzeug.

4.5.1.3.5 Delivery Tool

Das Delivery Tool schließlich stellt die eigentliche **Schnittstelle zum Benutzer** dar. Es ist für die Präsentation von **Abfrageergebnissen** und die Darstellung von aus der Wissensbasis **erzeugten Dokumenten** verantwortlich. Wie auch das Authoring Tool wird das Delivery Tool in Zusammenarbeit zwischen der Universität Koblenz-Landau und der Université de Nice entwickelt. Seine Funktionalität wird nachfolgend beschrieben (vgl. [TS final, 2003]):

- Es stellt eine **grafische Benutzungsschnittstelle** zur Interaktion mit dem Benutzer zur Verfügung.

[26] Siehe http://www.uni-koblenz.de/~dahn/

[27] Siehe http://www.unice.fr/

- Das **Generieren** von individuellen Dokumenten aus der Wissensbasis erfolgt über diese Benutzungsschnittstelle.
- Die automatische **Distribution** und der Austausch von Dokumenten und Lernmaterialien wird unterstützt.
- Zudem übernimmt das Tool die Aufgabe der **Autorisierung** und **Authentifizierung** des Benutzers.

4.5.1.3.6 Knowledge Management System

Das Knowledge Management System des Trial Solution Projekts erfüllt die Anforderungen an eine **Wissensverarbeitungskomponente** in einem wissensbasierten System (vgl. [RechPom, 1999], S. 978). Da das Ziel des Knowledge Management Systems die Generierung neuen Wissens auf der Basis gegebener bisheriger Entscheidungen des Benutzers (modelliert im Benutzermodell) ist, wurde eine **schichtenbasierte Architektur** für logische Programme als Grundlage verwendet (vgl. [TS final, 2003], S. 21 f.).

Aufgabe des Knowledge Management Systems ist es, aus den gegebenen Präferenzen und dem Vorwissen des Benutzers, den Lernzielen sowie den Metadaten der semantischen Einheiten in der Wissensbasis **Vorschläge zu generieren**, welche semantischen Einheiten in ein zu generierendes Dokument mit einzubeziehen sind (vgl. [TS final, 2003]).

4.5.2 Sharable Content Object Reference Model (SCORM)

SCORM wurde von der amerikanischen Organisation „Advanced Distributed Learning" (ADL)[28] ins Leben gerufen. Die **Zielsetzung** des Projekts ist ehrgeizig: es soll ein vollständiges Referenzmodell für die Erstellung von Wissensatomen und für die Implementierung von webbasierten Systemen, die Wissensatome und Lernmaterialien (Content) verwalten können, geschaffen werden. Bemühungen verschiedener weltweiter Organisationen zur Standardisierung auf dem Gebiet des e-Learning sollen dabei „unter einen Hut gebracht" werden (vgl. [Dodds, 2001a]).

Zur Illustration des SCORM Referenzmodells folgen zwei Abbildungen. Abbildung 34 zeigt, dass das SCORM Referenzmodell als eine **gemeinsame Basis unterschiedlicher Standards** zu verstehen ist. Jede(r) innerhalb von SCORM übernommene Standard oder Spezifikation wird als ein „Buch" inmitten einer „Bibliothek" betrachtet. In Abbildung 35 wird eine chronologisch geordnete Übersicht der Bemühungen und Arbeiten der ADL zur Vereinigung der Standards und Spezifikationen verschiedenster Organisationen angegeben.

[28] Siehe auch http://www.adlnet.org

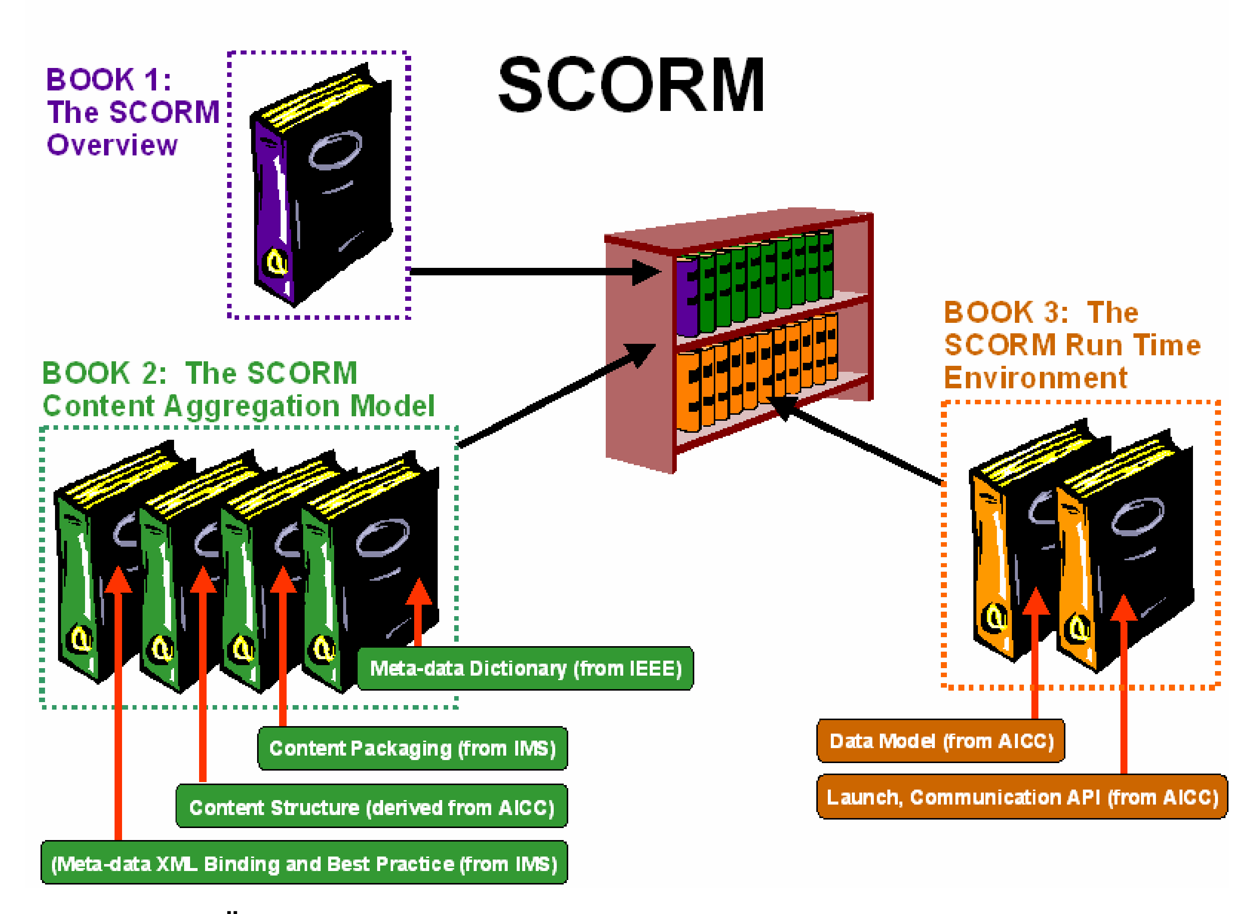


Abbildung 34: Übersicht über das SCORM Referenzmodell (Quelle: [Dodds, 2001a], S. „1-5")

Abbildung 35: SCORM als Sammlung diverser Standards und Spezifikationen (Quelle: [DoddsWest, 2002])

SCORM liegt gegenwärtig in Version 1.2 vor und umfasst folgende Teile:

- Content Aggregation Model (Abschnitt 4.5.2.2)
- Run-Time Environment Spezifikation (Abschnitt 4.5.2.3)

Im folgenden Abschnitt 4.5.2.1 wird zuerst ein Überblick über SCORM und die Designziele bei seiner Erstellung gegeben. Anschließend wird auf die beiden genannten Teile des Referenzmodells detailliert eingegangen.

4.5.2.1 Einleitung und Überblick über das Referenzmodell

Die Advanced Distributed Learning Initiative (ADL) wurde im November 1997 vom amerikanischen Verteidigungsministerium (*DoD – Department of Defense*) und dem *White House Office of Science and Technology Policy* (OSTP) gegründet. Zweck der ADL ist es, den Zugriff auf qualitativ hochwertige Materialien zur **Aus- und Weiterbildung** und zur **Entscheidungsunterstützung** sicher zu stellen. Wichtige Voraussetzung ist, dass die Materialien auf die Bedürfnisse eines Lernenden individuell anpassbar (personalisierbar) sind.

Durch die Entwicklung eines technischen **Frameworks** für die Implementierung von Systemen zur Verwaltung von Content und webbasierten Lernumgebungen wird die Stimulierung eines **Marktes für Lerninhalte** angestrebt. Das Framework erhielt den Namen „*Sharable Content Object Reference Model*" (SCORM). Als **Zukunftsvision** stellt sich die ADL eine Welt vor, in der Geräte für den Zugang zum Internet weit verbreitet und für jedermann erschwinglich sind sowie Bandbreiten in ausreichender Menge vorhanden sind. Folglich betreffen die Probleme bei der Schaffung zukünftiger e-Learning Umgebungen nicht das Vorhandensein einer Infrastruktur, sondern die **effiziente Nutzung** derselben.

Wenn erst einmal ein Markt für den Handel mit Lerninhalten ausreichend stimuliert, d.h. attraktiver gemacht, wurde, so die Ansicht der ADL, werden „**Wissensbibliotheken**" entstehen, in denen Wissensatome gesammelt und katalogisiert sind und von Nutzern der e-Learning Umgebungen erworben werden können. **Beide Seiten** werden vom Entstehen eines solchen Marktes **profitieren**: die Entwickler von Lernmaterialien werden sich um eine möglichst hohe Qualität der Inhalte bemühen, da in diesem Fall der zu erzielende Profit mit großer Wahrscheinlichkeit höher ist. Lernende ihrerseits können aus qualitativ hochwertigen Lernmaterialien einen höheren Lernnutzen ziehen (vgl. [Dodds, 2001a], S. „1-11" ff.).

Noch fehlen jedoch wesentliche Voraussetzungen zur Umsetzung dieser Vision. Die wichtigste Lücke besteht bei den **Standards**: ohne eine Standardisierung des Dateiformats von Lernmaterialien wird es nicht möglich sein, dass die Wissensatome eines Content-Herstellers innerhalb verschiedener Lernumgebungen verwendet werden können. Der Bedarf zur Entwicklung eines **gemeinsamen Frameworks** für die Erstellung von Lernmaterialien ergibt sich für die ADL aus folgenden Argumenten ([Dodds, 2001a], S. „1-7" ff.):

- Der rasante **Fortschritt** in Wissenschaft und Technik der vergangenen Jahre stellt eine große **Herausforderung** an Regierungen, Universitäten und Unternehmen in der Industrie dar, beinhaltet gleichzeitig aber enorme **Chancen**. Niemand, der im 21. Jahrhundert „am Ball bleiben" möchte, kann sich vor diesen Entwicklungen verstecken, sondern muss die Herausforderung annehmen und meistern.
- Trotz aller Technisierung bleiben der **Mensch und sein Wissen** der wichtigste Erfolgsfaktor. Der technische Fortschritt verlangt aber nach Arbeitskräften, die mit der neuen Technik vertraut sind. Verbesserungen beim Bildungsniveau der Menschen werden deshalb notwendig. Mit anderen Worten schafft die Entwicklung neuer Technologien einen **Fortbildungsbedarf**. Gleichzeitig werden aber auch die **Möglichkeiten** verbessert, die sich für die Gestaltung der Lehre und somit für die Deckung des erhöhten Fortbildungsbedarfs bieten.
- Selbst bei sorgfältiger Berücksichtigung der individuellen Leistungsfähigkeit können kaum homogene Klassen für den Gruppenunterricht gebildet werden. Die wertvollste Form des Unterrichts ist der **Individualunterricht**, bei dem ein Schüler jeweils von einem Betreuer unterrichtet wird. Nehmen Lehrer die Rolle des Betreuers ein, so ist die Forderung nach möglichst häufigem Einsatz von Individualunterricht nicht finanzierbar. Übernähmen Computer die Rolle des Betreuers, so könnte wesentlich besser und flächendeckender auf Individualunterricht zurück

gegriffen werden. Es ließe sich auf diese Weise die **Effizienz** des Unterrichts signifikant erhöhen.

- Studien haben bewiesen, dass durch die interaktive Verwendung des **Computers** die Effizienz des Unterrichts um 30 – 60 Prozent erhöht werden kann. Als Messgröße wurden die Kosten heran gezogen, die für die Erreichung einer Gesamtheit von Lernzielen in mehreren Versuchsgruppen anfielen (vgl. [Fletcher, 2001], zitiert in [Dodds, 2001a], S. „1-18" f.).
- Der Nutzen von computerbasiertem Unterricht kann weiter erhöht werden, wenn **asynchrone Lerntechnologien** zum Einsatz kommen. Diese Technologien vermitteln Wissen orts- und zeitunabhängig, d.h. der Lernende kann zu jeder Zeit an jedem Ort das Kursmaterial bearbeiten.
- Um zu gewährleisten, dass Lernmaterialien universell verwendbar bleiben, müssen diese **interoperabel** sein. D.h. Wissensatome sollen in beliebigen Lernumgebungen lauffähig bzw. verwendbar sein, unabhängig davon, von welchem Hersteller mit welchem Werkzeug diese erzeugt wurden. Daraus ergibt sich die Notwendigkeit zur Standardisierung auf dem Gebiet des e-Learning.
- Hersteller benötigen jedoch gemeinsame und verbindliche **Richtlinien**, nach denen sie ihre Wissensatome erzeugen können. Als Richtlinien kommen internationale oder nationale Standards, Spezifikationen, bewährte Praktiken, Empfehlungen oder Quasi-Standards in Frage. Zur erfolgreichen Formulierung und Anwendung der Richtlinien müssen diese auf einem gemeinsamen, allgemein akzeptierten **Referenzmodell** basieren.

Folglich ergibt sich die Notwendigkeit, ein Referenzmodell zu schaffen. ADL nennt drei **Hauptkriterien**, die ein solches Referenzmodell erfüllen muss [Dodds, 2001a].

a) Wie bereits oben erwähnt, muss das Referenzmodell die **Formulierung von Richtlinien** zu ihrer Anwendung unterstützen. Richtlinien sind dafür gedacht, bei der Produktion von Wissensatomen und der Implementierung von e-Learning Umgebungen konsultiert zu werden.

b) Das Referenzmodell muss von so vielen „Stakeholders" wie möglich **verstanden, akzeptiert und angewendet** werden. Besonders wichtig ist dies bei den Herstellern der e-Learning Umgebungen. Der Begriff Stakeholder meint alle Interessensgruppen, die an einem zukünftigen Markt für Lerninhalte teilnehmen (das sind Hersteller von Inhalten, Hersteller von Lernumgebungen, Konsumenten der Lerninhalte, usw.).

c) Die **Abbildung von spezifischen Standards** einzelner Interessensgruppen in das gemeinsame Referenzmodell muss nachvollziehbar sein, so dass jene, die zum gemeinsamen Modell beitragen, ihren jeweiligen Beitrag dazu erkennen können. Zudem muss ein proprietärer Standard auf das Referenzmodell zugeordnet werden können.

An die **Wissensatome** selbst ergeben sich die nachfolgend aufgezählten Hauptanforderungen („high level requirements") [Dodds, 2001a]:

- **Dauerhaftigkeit** (*durability*): wenn neue Versionen der Softwaresysteme entwickelt werden, darf dies nicht eine Veränderung der Wissensatome notwendig machen.
- **Interoperabilität** (*interoperability*): Wissensatome müssen auf einer Vielzahl von Hardware-, Betriebssystem- und Browserplattformen lauffähig und verwendbar sein.
- **Verfügbarkeit** (*accessibility*): die Wissensatome müssen indizierbar und durchsuchbar sein. Verschiedenartige Suchkriterien müssen angewendet werden können.
- **Wiederverwendbarkeit** (*reusability*): Wissensatome müssen durch eine Vielzahl von Entwicklungswerkzeugen modifizierbar und benutzbar sein.

Die präsentierten Überlegungen führten zur Entwicklung und Veröffentlichung des SCORM Referenzmodells. ADL sieht somit SCORM als ein Mittel zum Zweck, um folgende Ziele zu erreichen:

- Computer- und webbasierte Lernumgebungen **zu jeder Zeit**, und
- ...**an jedem Ort** zur Verfügung zu haben,
- ...wobei die Anpassung an **individuelle Bedürfnisse** wichtig ist.

Im englischen Original werden dafür die drei Schlagwörter **anytime**, **anywhere** und **tailored** genannt [Dodds, 2001a].

SCORM ist ein Referenzmodell für die Erstellung von **Wissensatomen bzw. Contentpaketen** (das sind Zusammenstellungen eines Kursmaterials oder Nachschlagewerks aus Wissensatomen), die über Systemgrenzen hinweg austauschbar sind. Es wird also ein **Datenmodell** definiert, das Interoperabilität zwischen verschiedenen e-Learning Umgebungen sicherstellen hilft. Außerdem stellt das Referenzmodell ein **Application Programming Interface (API)** zur Verfügung, mit dessen Hilfe Kommunikation zwischen den Contentpaketen und e-Learning Umgebungen realisiert werden kann. Mehr Details zu diesem Punkt folgen im Abschnitt 4.5.2.3 (Run-Time Environment Spezifikation, Seite 117 ff.).

4.5.2.2 Content Aggregation Model (CAM)

Das SCORM Content Aggregation Model (CAM) stellt für Designer und Programmierer von Lernumgebungen und Lernmaterialien ein Mittel dar, um Wissensatome (Lerninhalte, im englischen Originalbegriff „*learning resources*") zu „**Lernunterlagen**" zu bündeln (vgl. [Dodds, 2001b]). Der Begriff Lernunterlagen wird hier im Sinne von Kursmaterialien und Nachschlagewerken der Scholion WB+ Wissenstransfer-Anwendung verwendet.

Wissensatome sind die kleinsten **selbständigen Elemente** (Bestandteile, „Bausteine") einer Lernunterlage. Sie stellen die digitale Repräsentation jeder Art von Information dar, die im Rahmen eines Unterrichts verwendet werden und Wissen transportieren können. Ihre Existenz ist notwendig, um Lernunterlagen überhaupt **dynamisch** zu erstellen aus digitalen Inhalten, welche so abgespeichert sind, dass sie mehrfach verwendet werden

können. Im SCORM Referenzmodell wird dafür die „**Legostein-Metapher**" verwendet, welche in der folgenden Abbildung 36 veranschaulicht dargestellt ist.

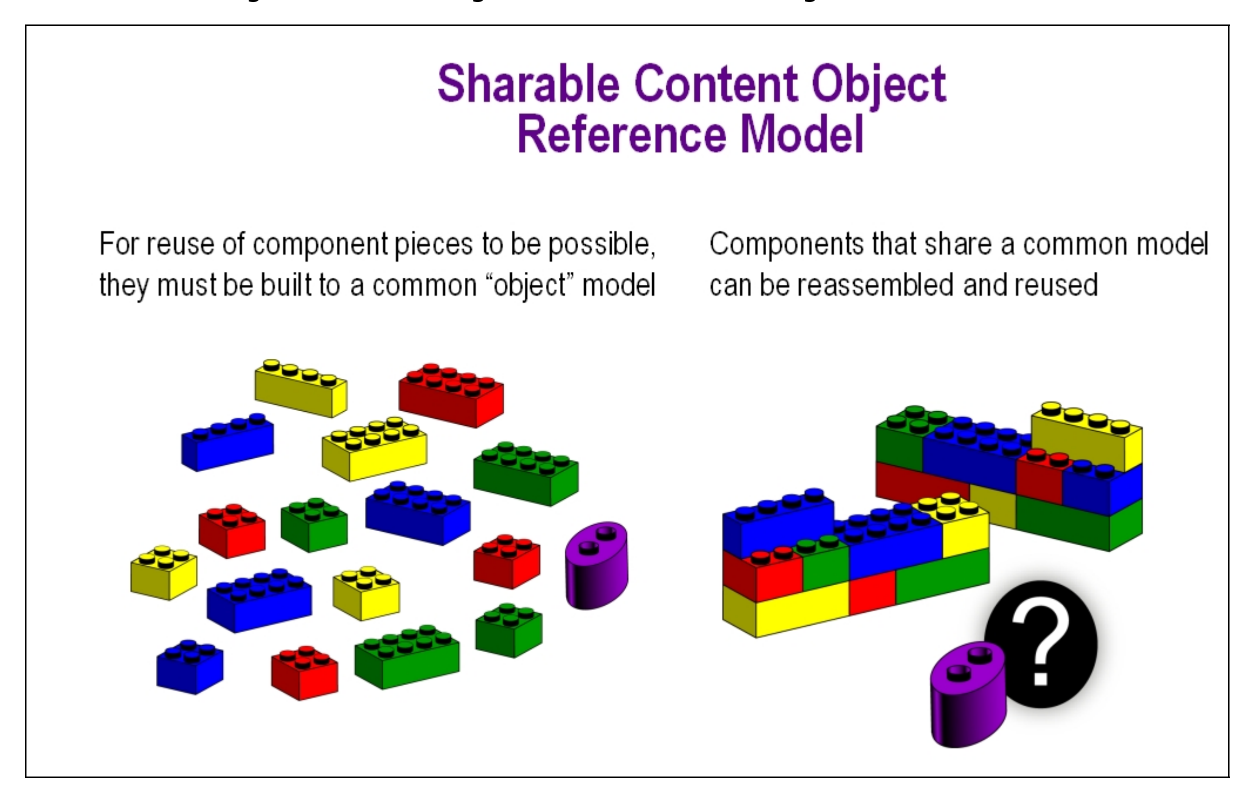


Abbildung 36: „Legostein-Metapher" des SCO Referenzmodells [DoddsWest, 2002]

Die wesentlichen, durch das CAM abgedeckten Aktivitäten für die Prozesse zur Erstellung und Nutzung von Lernunterlagen sind folgende [Dodds, 2001b]:

- **Erstellung** von Wissensatomen
- **Suche** nach Wissensatomen
- **Zusammenstellung** von Wissensatomen

Auf der Grundlage der genannten Aktivitäten ist die Erstellung von **komplexeren Objekten** und letztlich Lernunterlagen aus den **einfachen Bausteinen**, den Wissensatomen, möglich. Die Aktivitäten erfordern die Definition folgender Teilmodelle des Content Aggregation Modells [Dodds, 2001b]:

- **Content Model**: dies umfasst Konzepte für die **Abbildung und Speicherung** von digitalen Ressourcen beliebiger Art und Größe in einer Wissensbasis sowie ein Begriffssystem (Nomenklatur) für die Bausteine, die zur Zusammenstellung von Lernunterlagen verwendet werden.
- **Meta-data**: ein Mechanismus für die Beschreibung der Lernunterlagen einerseits und der Bausteine einer Lernunterlage andererseits ist unerlässlich, um eine **gezielte Suche** nach bestimmten Inhalten zu ermöglichen. Dieser Mechanismus ist am Einfachsten mit Daten über die Inhalte, oder „Metadaten", realisierbar.

- **Content Packaging**: das „Verpacken" von Inhalten in Form von Wissensatomen umfasst zwei Aspekte. (1) Das **interne Verhalten** einer Lernunterlage (die Struktur des Inhaltes – Content Structure) betrifft vor allem Aspekte der Präsentation von Inhalten an die Lernenden; und (2) das **Format** eines Pakets (Content Package – d.h. einer Datei) mit Lernunterlagen, welches für den **Austausch** von Lernunterlagen zwischen verschiedenen Systemen große Bedeutung besitzt.

Im Rahmen der folgenden Punkte werden die genannten Teilmodelle des CAM erläutert. Punkt 4.5.2.2.1 geht auf die Beschreibung der Komponenten des SCORM Content Model ein. In Punkt 4.5.2.2.2 wird dagegen untersucht, in welcher Form und auf welchen Granularitätsniveaus Inhalte aller Art im SCO Referenzmodell mit Metadaten versehen werden. Schließlich beschäftigt sich Punkt 4.5.2.2.3 mit der Erläuterung von Konzepten zur Bildung von Lernunterlagen (Content Packages).

4.5.2.2.1 Komponenten des SCORM Content Model

Das SCORM Content Model umfasst alle Aspekte, die die Objekte betreffen, welche für die Erstellung einer Lernunterlage aus wiederverwendbaren Wissensatomen (Learning Objects) benötigt werden. Es werden zuerst **Granularitätsniveaus** definiert, in denen Inhalte als Wissensatome oder Lernunterlagen abgespeichert werden können. Zudem sind **Regeln** festgelegt, wie Objekte eines Granularitätsniveaus zu Objekten des nächst höheren Niveaus **zusammen gesetzt** werden dürfen.

Im SCORM werden folgende Komponenten (diese entsprechen jeweils einem Granularitätsniveau) definiert:

- 4.5.2.2.1.1 Assets (Inhaltsbrocken)
- 4.5.2.2.1.2 Sharable Content Objects – SCOs
- 4.5.2.2.1.3 Content Aggregations (Inhaltspakete oder Lernunterlagen)

Es folgen Erläuterungen zu den genannten Komponenten.

4.5.2.2.1.1 Assets (Inhaltsbrocken)

Assets oder auch „Inhaltsbrocken" sind die kleinsten „Körnchen" oder „Splitter", in denen Lerninhalte dargestellt sein können. Ein *Asset* ist jede **digitale Repräsentation** von Information, sei es in Form von Text, Bild, Ton (Audiofile), einer Webseite, Assessment Objekten oder vielem mehr. Abbildung 37 zeigt eine schematische Darstellung möglicher Beispiele für *Asset*-Objekte. *Assets* können mit Hilfe von Metadaten näher beschrieben werden, so dass eine Suche innerhalb der gespeicherten Inhaltsbrocken möglich wird.

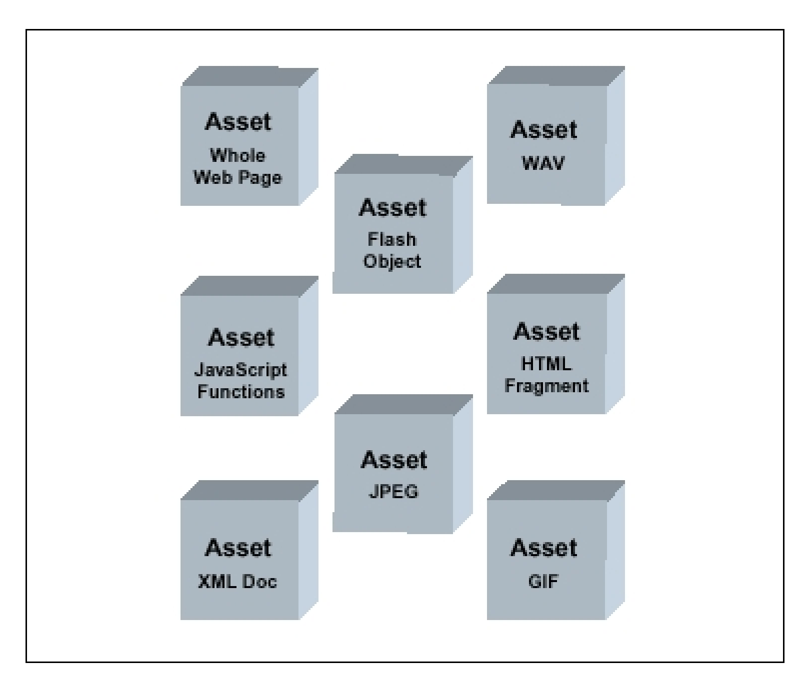


Abbildung 37: Beispiele für SCORM Assets (Inhaltsbrocken) [Dodds, 2001b], S. „2-4"

4.5.2.2.1.2 Sharable Content Objects – SCOs

Ein *Sharable Content Object* (SCO, „Wissensatom") im SCORM Referenzmodell ist eine **Sammlung** aus einem oder mehreren Inhaltsbrocken (*Assets*), von denen eines ein spezielles ausführbares ist. Das **ausführbare Asset** übernimmt Aufgaben der Kommunikation des SCOs mit dem **SCORM Run Time Environment** (für eine Beschreibung siehe Abschnitt 4.5.2.3, Seite 117 ff.), wodurch sicher gestellt ist, dass eine Lernunterlage mit allen jenen Lernumgebungen (*Learning Management System LMS*) kommunizieren kann, die das SCORM Run-Time Environment unterstützen. Eine Illustration eines beispielhaften SCOs findet sich in Abbildung 38.

Aus der obigen Definition geht hervor, dass ein *Sharable Content Object* sinngemäß einem Wissensatom entspricht, wie sie im Contentpool von Scholion WB+ zu unterstützen sind. *Assets* dagegen sind ein noch feineres Granularitätsniveau, das in der Wissenstransfer-Umgebung Scholion WB+ nicht implementiert wird.

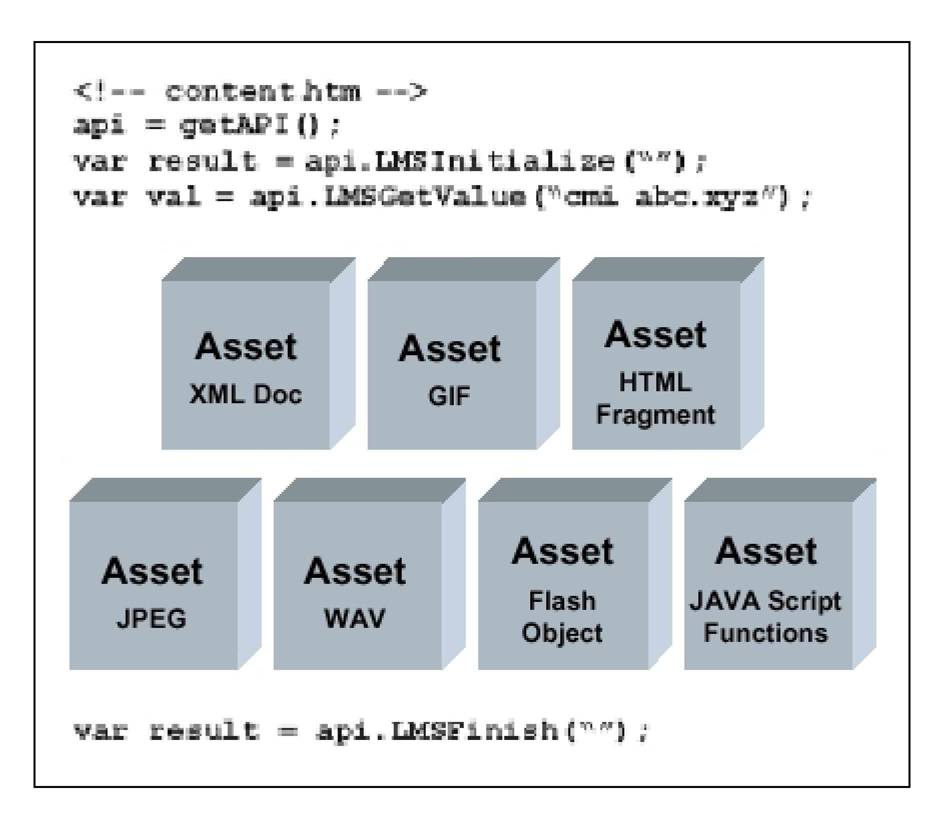


Abbildung 38: Beispiel eines SCOs (Wissensatoms) [Dodds, 2001b], S. „2-5"

Um eine effiziente Wiederverwendung zu gewährleisten, sollte innerhalb der SCOs **keinerlei Bezug** zu spezifischen Lernzielen genommen werden, da Lernziele fast ohne Ausnahme nur in Zusammenhang mit bestimmten Lernsituationen Gültigkeit besitzen. Ein *Sharable Content Object* des SCORM Referenzmodells ist das niedrigste Granularitätsniveau für *learning resources*, das mit der Schnittstelle des SCORM Run-Time Environments **„angesprochen"** und verwendet werden kann (vgl. [Dodds, 2001b]).

Die **Größe eines Wissensatoms** ist vom Referenzmodell nicht zwingend vorgegeben. Es sollte eine Größe gewählt werden, die die Wiederverwendbarkeit des Atoms maximiert. Aus dieser Forderung ergibt sich, dass ein SCO **so klein wie möglich**, aber **so groß wie nötig** sein muss – die semantische Eindeutigkeit muss erhalten bleiben. Ein Wissensatom muss mit anderen Worten eine in sich geschlossene Bedeutung besitzen, so dass zum eindeutigen Verständnis des gespeicherten Inhalts keine weiteren Information benötigt wird.

Wissensatome (SCOs) können ebenfalls mit **Metadaten** beschrieben werden, so dass eine Datenbank (oder sogar eine Wissensbasis) aus ihnen aufgebaut werden kann. Das **Suchen und Auffinden** von Wissensatomen für bestimmte Zwecke wird so ermöglicht. Das wirkt sich positiv auf die Wiederverwendbarkeit von Wissensatomen aus.

Es wurde bereits weiter oben erwähnt, dass ein SCO mit dem SCORM Run-Time Environment kommuniziert. Daraus ergibt sich, dass ein SCO die Fähigkeit besitzen muss, das **Application Programming Interface (API)** einer Lernumgebung (eines LMS) zu lokalisieren und anzusprechen. Außerdem ergibt sich daraus, dass SCOs nur von Lernumgebungen verwendet werden können, welche das SCORM Run-Time Environment in

ihrem API unterstützen. Daraus ergeben sich wesentliche Vorteile für die **Distribution und Nutzung** der Lernunterlagen, nämlich **Interoperabilität** zwischen verschiedenen Lernumgebungen und eine **Abstraktion** vom inneren Verhalten der Lernunterlage. Eine Lernumgebung kann folglich Lernunterlagen unterschiedlicher Hersteller in der selben Weise ansprechen und mit ihnen interagieren [Dodds, 2001b].

4.5.2.2.1.3 Content Aggregations (Inhaltspakete oder Lernunterlagen)

Lernunterlagen (Content Aggregations) dienen dazu, die Wissensatome zu **komplexeren „Bündeln"** zusammen zu schnüren. Die sinnvolle Zusammenstellung obliegt dabei dem Autor des Kursmaterials oder des Nachschlagewerks. Auch die Inhaltspakete können mit Metadaten beschrieben werden, so dass ihre Speicherung und Auffindung erleichtert wird. Abbildung 39 enthält die Darstellung eines Beispiels für eine SCORM Lernunterlage. Die Darstellung in dieser Grafik entspricht einer konzeptuellen Modellierung; die verwendeten Begriffe dürfen nicht mit den im Datenmodell verwendeten Begriffen verwechselt werden.

Das beschriebene Konzept stellt eine wesentliche Verbesserung gegenüber früherer, proprietärer Lösungen für die Speicherung von Lernunterlagen dar. Charakteristisch ist die **Trennung zwischen dem eigentlichen Inhalt** (den Wissensatomen oder SCOs) **und der Information zur Struktur**, die zwischen den Inhaltselementen herrscht. Zudem ist das SCO Referenzmodell vollständig webbasiert, wodurch der Zugang zu den Lernunterlagen und den darin verwendeten digitalen Ressourcen wesentlich erleichtert wird.

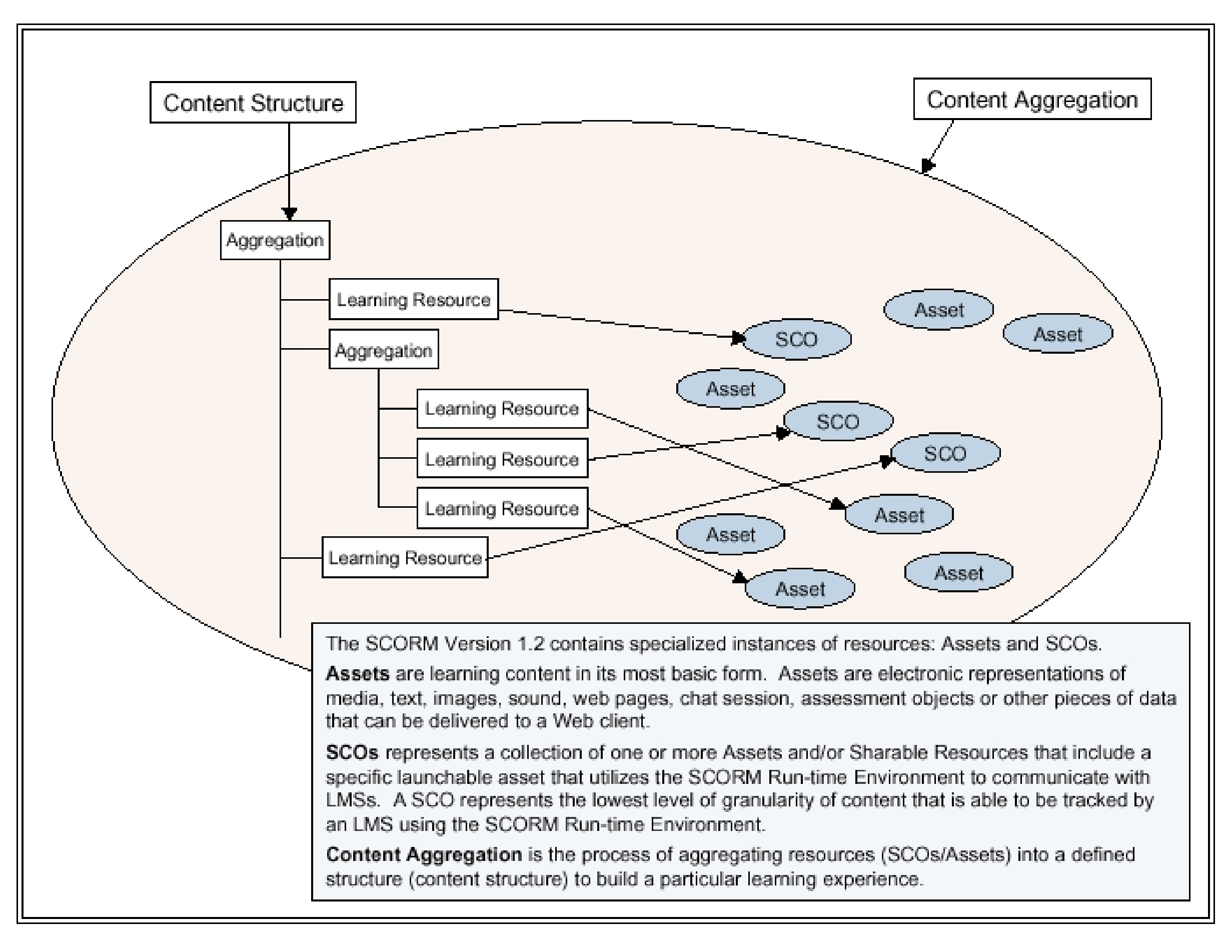


Abbildung 39: Darstellung eines SCORM Content Packages [Dodds, 2001b], S. „2-7"

Eine **wichtige Voraussetzung** für das Funktionieren des Konzepts sei nochmals erwähnt. Die Wiederverwendbarkeit eines Wissensatoms leidet wesentlich oder wird sogar ganz unmöglich, falls sein Inhalt nur in Abhängigkeit von anderen Wissensatomen **semantisch verständlich** und erfassbar ist. Das hätte zur Folge, dass das Wissensatom ohne seinen „Zwillingsbruder" nicht sinnvoll verwendet werden kann. Anhand der Abbildung 39 kann nun besser nachvollzogen werden, warum sich dies so negativ auswirkt. Wenn sich das zusätzlich benötigte Wissensatom außerhalb der Lernunterlage befindet (in der Abbildung außerhalb der Ellipse) und gegenwärtig nicht verfügbar ist (z.B. aufgrund eines Serverausfalls), so wird der praktische Wert der Lernunterlage unangenehm reduziert.

4.5.2.2.2 SCORM Metadaten

Metadaten sind **Daten über Daten**. Der Zweck von Metadaten ist die Vereinheitlichung der Beschreibung von Daten. Konkret geht es darum, einen Mechanismus zur Verfügung zu stellen, mit dessen Hilfe Wissensatome (*learning resources*) in einer **einheitlichen Weise beschrieben** werden können. Die Beschreibung von Wissensatomen dient dem Zweck, diese in einer Datenbasis abspeichern und auch wieder effektiv auffinden zu können. **Effektives Auffinden** von Wissensatomen bedeutet, die richtigen Wissensatome für einen spezifischen Verwendungszweck zu erhalten.

SCORM übernimmt für seine Metadaten die Spezifikationen des **IMS** Global Learning Consortiums. Sowohl das Metadata Information Model als auch das Metadata XML Binding wurden ohne Veränderungen übernommen. Beide Spezifikationen wurden bereits in diesem Dokument vorgestellt, weshalb an dieser Stelle auf eine ausführlichere Darstellung verzichtet wird. Details zu den IMS Metadaten Spezifikationen können im Abschnitt 4.3.3.1 (siehe Seite 58 ff.) nachgelesen werden.

Es wurde bereits im vorangegangenen Punkt 4.5.2.2.1 (Komponenten des SCORM Content Model) erwähnt, dass Metadaten auf **jedem Granularitätsniveau** der Inhalte mitgespeichert werden können. Konkret bedeutet dies [Dodds, 2001b]:

- Auf Ebene der **Lernunterlagen** (Kursmaterialien und Nachschlagewerke) oder Content Aggregations dienen Metadaten zum Auffinden von Unterlagen innerhalb einer elektronischen Bibliothek. Dazu müssen z.B. die Lernziele oder Daten über die Lernsituation vorliegen.
- Auf **Wissensatom-Ebene** (Ebene der SCOs) enthalten Metadaten beschreibende Information über die Inhalte, die in einer Datenbank oder einer Wissensbasis gespeichert sind. Auf dieser Ebene können Suchvorgänge für die Zusammenstellung von Kursunterlagen spezifiziert werden.
- Auf der untersten Ebene, jener der **Assets** (Inhaltsbrocken), werden die Metadaten zu den Rohmaterialien für die Lernunterlagen, unabhängig von jedwedem Bezug auf einen Einsatzzweck, gespeichert. Diese Metadaten werden zur Zusammenstellung von Wissensatomen benötigt.

Für die Metadaten zu allen Ebenen wird die IMS Metadaten Spezifikation verwendet. Es liegt jedoch auf der Hand, dass je nach Verwendungszweck manche Elemente der spezifizierten Metadaten-Kategorien nicht zwingend benötigt werden. Das SCORM Referenzmodell definiert hierzu eine Übersicht von **Anwendungsrichtlinien** (Application Profiles), die für jede der Granularitätsniveaus die Metadaten-Kategorien entweder als verpflichtend oder als optional markiert. Eine Wiedergabe der Details dazu geht über den Rahmen dieses Abschnitts hinaus; der Leser wird auf die Quelle verwiesen ([Dodds, 2001b], S. „2-96" ff.).

4.5.2.2.3 SCORM Content Packaging

Content Packaging ist der Sammelbegriff für Konzepte, die dem **standardisierten Austausch** von digitalen Lernunterlagen (Ressourcen für Lernzwecke – „digital learning resources") dienen. Wie in der Einleitung zum Abschnitt 4.5.2.2 (Content Aggregation Model (CAM)) auf Seite 107 erläutert, umfassen diese Konzepte zwei verschiedene Aspekte:

- Struktur und internes Verhalten des Contentpakets
- Format, in dem das Paket und die enthaltenen Ressourcen distribuiert (ausgetauscht) werden

Das SCORM Referenzmodell verwendet *Content Packages*, um Lernunterlagen (Kursmaterialien und Nachschlagewerke) in digitaler Form zwischen Lernumgebungen (Lernmanagementsystemen LMS), Entwicklungswerkzeugen und digitalen Wissensbasen austauschen zu können. Die **Strukturinformation** zu *Content Packages* kann als „Landkarte"

verstanden werden, mit deren Hilfe der Lernende durch das Paket geführt wird, also die Ressourcen durchwandert werden können. Durch das vorgegebene Format des SCORM *Content Packaging* werden folgende Aspekte definiert:

- Eine **„Packliste" (manifest file)** zur Beschreibung des Content-Pakets; diese enthält Metadaten über das Package, einen optionalen Abschnitt zur Beschreibung der Struktur und des internen Verhaltens des Pakets sowie eine Liste der Referenzen zu den eigentlichen Ressourcen.
- Anweisungen, wie ein Manifest auf Basis von XML zu erstellen ist, d.h. eine **XML-Spezifikation** für die Manifest-Datei.
- Richtlinien für die **Speicherung** des Manifestes zusammen mit den physischen Ressourcen in distribuierbarer (austauschbarer) Form, z.B. in einer Zipdatei oder auf einem Datenträger.

Für die Umsetzung der genannten Definitionen verwendet SCORM das IMS Content Packaging Information Model in beinahe unveränderter Form. Die Spezifikation der Umsetzung des Information Model auf Basis von XML wurde ebenfalls vom IMS Global Learning Consortium übernommen (IMS Content Packaging XML Binding Specification). Beide genannten Spezifikationen wurden in diesem Dokument schon behandelt, weshalb an dieser Stelle auf eine genauere Darstellung der Details verzichtet wird. Ausführlichere Beschreibungen dazu können im Abschnitt 4.3.3.2 (siehe Seite 63 ff.) nachgelesen werden. In diesem Abschnitt erfolgt lediglich eine Darstellung der Erweiterungen dieser Spezifikationen und Hinweise auf wichtige Besonderheiten.

Die folgende Abbildung 40 zeigt das konzeptuelle Modell eines Content Packages gemäß der IMS Spezifikation. (**Hinweis**: vgl. dazu auch Abbildung 16, Seite 63.)

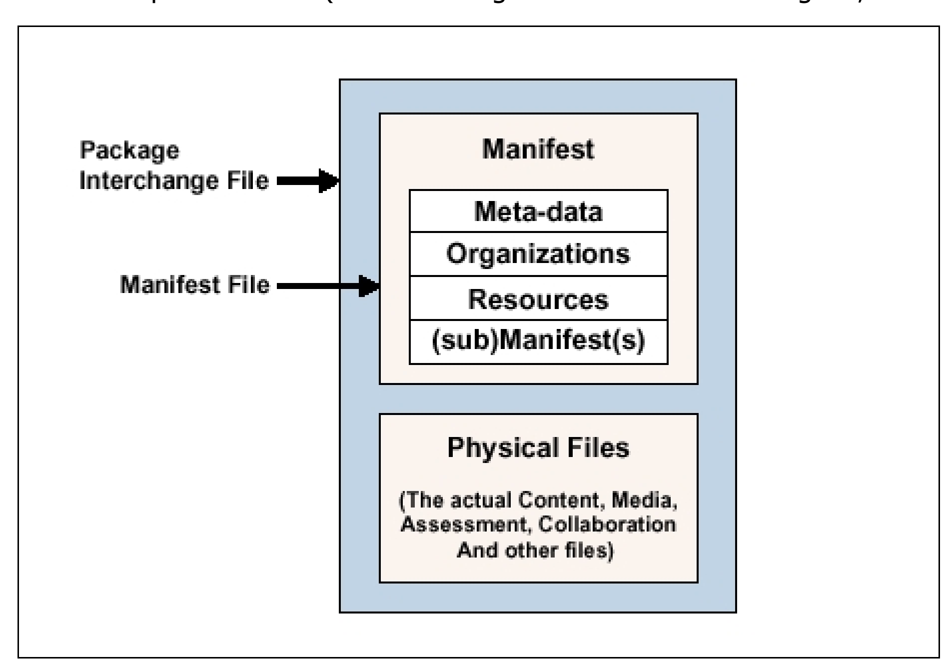


Abbildung 40: Konzeptuelles Modell – SCORM Content Packaging (Quelle: [Dodds, 2001b], S. „2-111")

Erweiterungen von SCORM zu den IMS Spezifikationen betreffen zunächst vor allem Richtlinien für die Anwendung. SCORM sieht vor, dass Inhalte auf allen Granularitätsniveaus (das sind *Assets*, *SCOs* und *Content Aggregations*) in ein *Content Package* verpackt werden dürfen. Für jedes dieser Objekte gelten spezifische Richtlinien und Empfehlungen. Darüber hinaus wurde das Datenmodell um einige zusätzliche Elemente erweitert, welche in der Quelle [Dodds, 2001b] (Seite „2-101" ff.) explizit angegeben sind.

Die wichtigste Erweiterung, die im SCO Referenzmodell vorgenommen wurde, betrifft eine genauere Definition des „**Organizations**" Elements im IMS Content Package. Dieses Element kann dazu verwendet werden, um **empfohlene Navigationspfade** (d.h. Präsentationsreihenfolgen) der Inhalte zu definieren. ADL verwendet dazu im Rahmen von SCORM eine vom AICC definierte Skriptsprache, nämlich „**aicc_script**" [AICC, 2001]. Mehr Details dazu folgen weiter unten.

In der Folge werden alle wichtigen Aspekte erläutert, die mit der Struktur eines Content Packages in Zusammenhang stehen. Diese sind:

- Content Hierarchy (Inhaltsstrukturierung)
- Context Specific Meta-data (kontextabhängige Metadaten)
- Sequencing and Navigation (Navigationspfade)

4.5.2.2.3.1 Content Hierarchy (Inhaltsstrukturierung)

Die Hierarchie (oder Strukturierung) der Inhalte eines Content Packages kann als **Inhaltsverzeichnis** für das Paket betrachtet werden. Es handelt sich dabei um eine **Baumrepräsentation** der im Paket gespeicherten Ressourcen. Meist wird damit die vom Autor empfohlene Standard-Präsentationsreihenfolge abgebildet, es kann sich jedoch auch um eine völlig willkürliche Strukturierung handeln.

Das konzeptuelle Modell für die Erstellung einer Strukturierung wird in der folgenden Abbildung 41 gezeigt.

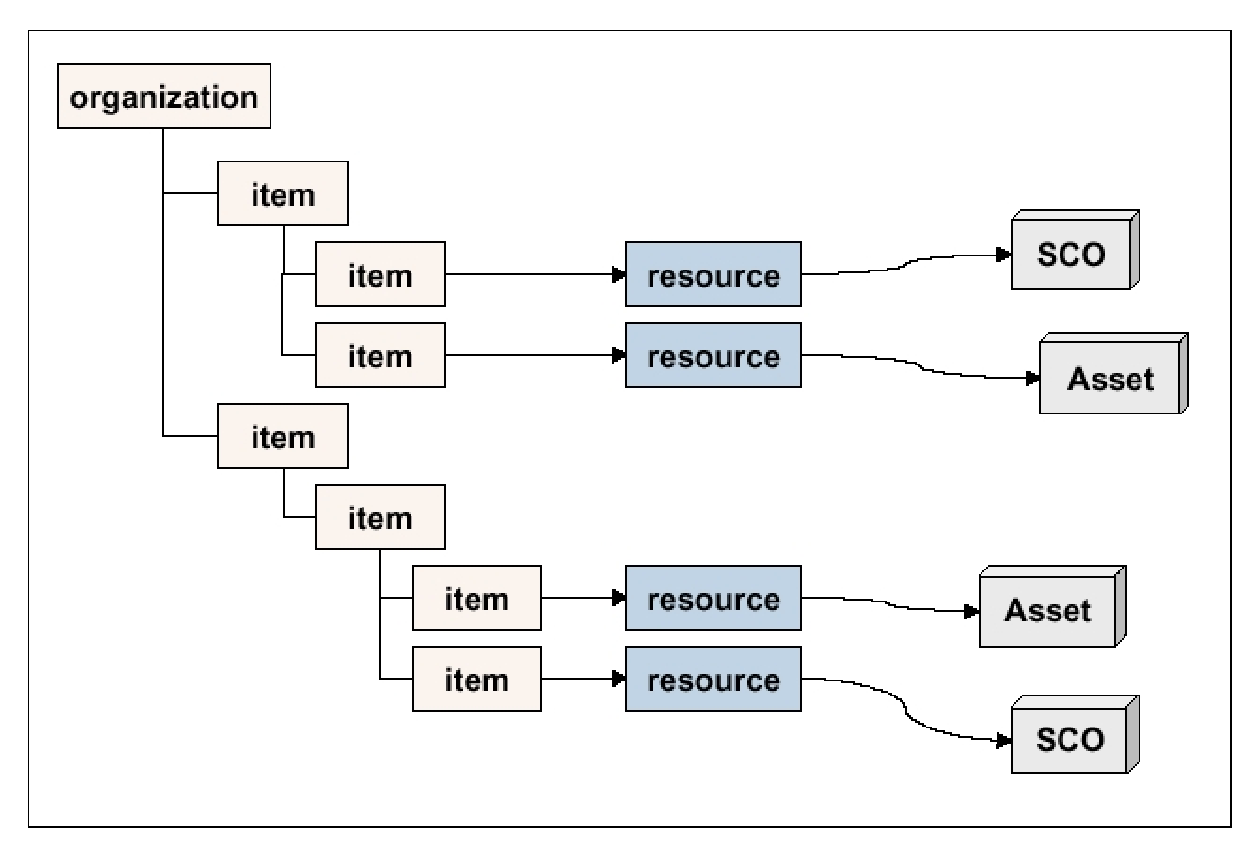


Abbildung 41: SCORM Hierarchisierung von Content Packages (Quelle: [Dodds, 2001b], S. „2-106")

Hierarchien für die Inhalte werden sowohl für das „Organizations" als auch für das „Resources" Element eines *Content Packages* definiert (vgl. Abbildung 40, Seite 113). Mit Hilfe einer Strukturierung der Inhalte können Konzepte wie die Einteilung der Lernunterlage in Kurse, Kapitel, Themen und andere Einheiten verwirklicht werden.

4.5.2.2.3.2 Context Specific Meta-data (kontextabhängige Metadaten)

Metadaten auf Ebene der *Content Packages* dienen dazu, die empfohlene **Lernstrategie** für die Abarbeitung einer Lernunterlage, zusätzliche **Beschreibungen** der enthaltenen Wissensatome oder Hinweise zur Anwendung des Wissens anzugeben. Information dieser Art wird in einem *Content Package* mitgespeichert und mitgeliefert (vgl. Abbildung 40, Seite 113). Sie dient zur leichteren Auffindung der richtigen Lernunterlage auf Basis eines vorgegebenen Lernziels oder ähnlicher Daten.

Hinweis: Im Gegensatz dazu gibt es auch kontextunabhängige Metadaten, welche auf Ebene der Wissensatome abgespeichert werden und dazu dienen, die passenden Atome für die Erstellung einer Lernunterlage zu finden. Darauf wurde bereits im Punkt 4.5.2.2.2 hingewiesen.

4.5.2.2.3.3 Sequencing and Navigation (Navigationspfade)

Die Regeln, die von einer Lernumgebung (LMS) bei der Präsentation der Wissensatome aus einem *Content Package* eingehalten werden müssen, werden Navigationspfade genannt. Es liegt im Verantwortungsbereich des Designers von Lernunterlagen, solche Pfa-

de zu definieren und in einem *Content Package* abzuspeichern. In der Regel werden Reihenfolgen definiert, die nach Ansicht des Autors den **größten Lerneffekt erzielen**. Die Speicherung der definierten Pfade erfolgt im „Organizations"-Element des *Content Packages* (vgl. Abbildung 40, Seite 113).

An dieser Stelle kommt im SCO Referenzmodells die vom AICC ausgearbeitete Skriptsprache „aicc_script" ins Spiel. Das IMS Content Packaging Information Model erlaubt für jedes Item (vgl. Abbildung 41, Seite 115) die Definition von so genannten **Lernvoraussetzungen („prerequisites")**. Das sind Verweise auf andere Items, die als Voraussetzungen zum Verständnis des ausgewählten Items gelten. Die Modellierung der Verweise wird mit den Operatoren von aicc_script vorgenommen. In der nachfolgenden Tabelle 2 sind alle Operatoren dieser Sprache angegeben.

Tabelle 2: Zulässige Operatoren der AICC Skriptsprache „aicc_script" [AICC, 2001]

<table>
<tr><th>Operator</th><th>Description and examples</th></tr>
<tr><td>And
&</td><td>All elements separated by an & must be complete for the expression to be evaluated as complete.
S34 & S36 & S38
SCOs number S34, S36 & S38 must all be complete ("passed" or "completed" for the group to be considered complete.</td></tr>
<tr><td>Or
|</td><td>If any of the elements separated by an | are complete ("passed" or "completed") the expression is considered true.
S34="passed" | S36="passed" | S38="passed"
If any one of the SCOs S34, S36, or S38, are passed then the group is considered complete.</td></tr>
<tr><td>Not
~</td><td>An operator that returns incomplete (false) if the following element or expression is complete, and returns complete (true) if the follwowing element or expression is incomplete (false).
Element Identifier: S34
Requirement: ~S35
The student may enter SCO S34 as long as SCO S35 has not been completed (that is, the status of S35 must be "incomplete", "failed" or "not attempted"). If SCO S35 is complete, the student may not enter SCO S34.</td></tr>
<tr><td>Equals
=</td><td>An operator that returns true when representations on both sides of the symbol have the same values.
Element identifier: S34
Requirement: S33="passed"
The student may enter SCO S34 if he has passed SCO S33.</td></tr>
<tr><td>Not equals
<></td><td>An operator that returns true when elements on both sides of the symbol have different values.
Element identifier: S34
Requirement: S33<>"passed"
The student may enter SCO S34 as long as he or she has not passed SCO S35. Notice the difference between this expression and the example for the not operator. The equivalent of ~S35 is (S35<>"passed" & S35<>"completed").</td></tr>
</table>

Set {}	A list of learning content elements (SCO or Block) separated by commas and surrounded by curly brackets – { }. A set differs from a Block, in that the set is defined only for purposes of the prerequisite file. A set has no effect on the structure of the learning content. {S34, S36, S37, S39} SCOs S34, S36, S37 and S39 are part of a set.
Separator ,	The comma is used to separate the members of a set. Each member of the set can be evaluated as a Boolean element – complete or incomplete. {S34, S36, S37, S39} SCOs S34, S36, S37 and S39 are each separated by a comma in this set.
X*	X is an integer number. This operator means that X or more members of the set that follows must be complete for the expression to be complete (true). Element identifier: S38 Requirement: 3*{S34, S36, S37, S39} Any three or more of the following SCOs – S34, S36, S37, S39 – must be complete ("passed" or "completed") before the student can enter SCO S38.
Precedence ( )	The expression inside the parenthesis ( ) must be evaluated before combining its results with other parts of the logical statement. Parentheses may be nested. (Operator precedence is the same as in the C programming language – including the use of parenthesis.) Element identifier: S39 Requirement: S34 & S35 | S36 In this statement, completing S36 all by itself enables the student to enter S39. Element identifier: S39 Requirement: S34 & (S35 | S36) Adding the parenthesis, makes it necessary to complete ("passed" or "completed") at least two units (S36 all by itself is no longer enough) to enter unit S39.
Vocabulary for all elements	• passed • completed • browsed • failed • not attempted • incomplete

Hinweis: Bis jetzt wurden die Aspekte zur Abbildung der Struktur eines *Content Packages* behandelt, ohne noch auf das **interne Verhalten** einzugehen. Dieses wird nicht auf Ebene der Lernunterlagen, sondern auf Ebene der wiederverwendbaren Wissensatome gespeichert. Die Daten eines *Content Packages* werden an die Lernumgebung (das LMS) weitergegeben; für dieses ist jedoch nur die Struktur in Verbindung mit Information zur Präsentation der Inhalte interessant – wie, d.h. mit welchen Mitteln und Medien und in welcher Codalität letztendlich die Präsentation erfolgt (mit anderen Worten, das interne Verhalten der Lernunterlage), ist Angelegenheit eines jeden Wissensatoms selbst.

4.5.2.3 Run-Time Environment Spezifikation

Wesentlicher Bestandteil von SCORM ist eine Laufzeitumgebung, die benötigt wird, um das Ziel von **Interoperabilität** von Lernunterlagen zwischen verschiedenen Systemen zu erreichen. Wenn ein Kursmaterial oder ein Nachschlagewerk in verschiedenen Lern-

umgebungen (LMS) lauffähig sein soll, so muss es ein standardisiertes Vorgehen zum **Starten von Kursen**, einen einheitlichen **Kommunikationsmechanismus** für die Interaktion von Lernumgebung und Lernunterlage und eine **vordefinierte Sprache** oder zumindest ein Vokabular für die Kommunikation geben. Alle diese drei Aspekte werden vom SCORM Run-Time Environment berücksichtigt und definiert, wie es Abbildung 42 veranschaulicht.

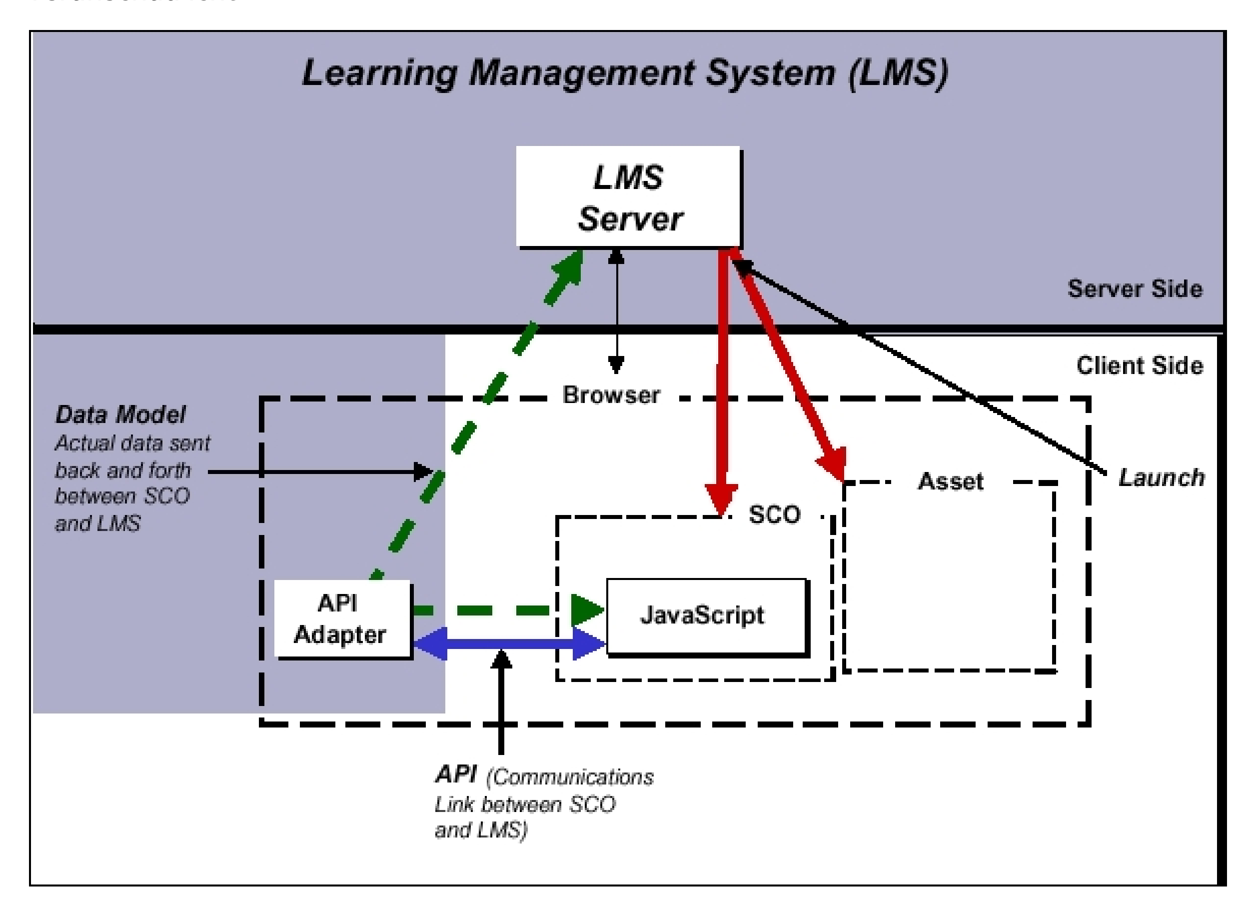


Abbildung 42: Konzeptuelles Modell – SCORM Run-Time Environment [Dodds, 2001c]

Dementsprechend unterteilt sich dieser Abschnitt in die Präsentation der drei genannten Komponenten [Dodds, 2001c]:

- **4.5.2.3.1 Launch mechanism (Startmechanismus)**: definiert die Prozeduren und Verantwortlichkeiten für die Initialisierung (Konfiguration) der Kommunikation zwischen der Lernunterlage und der Lernumgebung.
- **4.5.2.3.2 Application Programming Interface (API)**: das API ist der Mechanismus zur Kommunikation der Lernunterlage mit der Lernumgebung, z.B. um Mitteilungen zum gegenwärtigen Abarbeitungsstatus zu geben oder um Daten zwischen der Lernumgebung und den Wissensatomen austauschen zu können.
- **4.5.2.3.3 Data Model (Datenmodell)**: eine standardisierte Menge von Datenelementen, durch welche die auszutauschenden Information nach Art (Datentyp) und Menge (Anzahl) definiert werden.

In der Folge werden die oben aufgezählten Komponenten detailliert erläutert.

4.5.2.3.1 Launch mechanism (Startmechanismus)

Ein standardisierter Startmechanismus ist die Voraussetzung für **Konsistenz im Verhalten** von Lernunterlagen, die in verschiedenen Wissenstransfer-Umgebungen gestartet werden, unabhängig von der Implementierung der verwendeten Lernumgebung.

Wie bereits im Abschnitt 4.5.2.2.1 (siehe Seite 107 ff.) beschrieben wurde, besteht das SCORM *Content Aggregation Model* (CAM) aus drei Komponenten:

- *Asset*
- *SCO*
- *Content Aggregation*

Von einer Lernumgebung (einem Lernmanagementsystem LMS) können **lediglich Assets und SCOs gestartet** werden. *Content Aggregations* müssen vor Verwendung durch das LMS „entpackt" werden. Anschließend daran können die im Package enthaltenen Objekte (*Assets* und *SCOs*) separat gestartet werden [Dodds, 2001c].

Das **Management der Navigation** durch eine Lernunterlage liegt in der Verantwortlichkeit des LMS. Mit Hilfe der im *Content Package* gelieferten Strukturinformation und empfohlenen Navigationspfade können **Hinweise auf eine sinnvolle Präsentationsreihenfolge** gewonnen werden. Empfehlungen, die im *Content Package* enthalten sind, müssen jedoch keineswegs eingehalten werden. Das LMS könnte auch durch die Berücksichtigung von Daten (z.B. über Vorkenntnisse und bereits erzielte Lernfortschritte) des Lernenden zu anderen Navigationspfaden kommen. Auch eine vom Benutzer definierte Steuerung durch die Lernunterlage ist denkbar.

Das LMS ist ebenfalls dafür verantwortlich, dass die ausgewählten Wissensatome auch tatsächlich in der **vorgesehenen Reihenfolge gestartet** werden. Zudem muss sichergestellt sein, dass immer nur **ein Wissensatom zugleich** in den aktiven Zustand geschaltet ist.

Je nachdem, ob ein *Asset* (Inhaltsbrocken) oder ein *SCO* (Wissensatom) gestartet wird, ergeben sich noch einige Besonderheiten (vgl. [Dodds, 2001c]).

- **Assets** müssen vom LMS lediglich gestartet werden, benötigen jedoch keine Information (bzw. Rückmeldungen) vom LMS. Dadurch ist eine weitere Kommunikation nicht mehr notwendig.
- **SCOs (Wissensatome)** dürfen ausschließlich **sequentiell** gestartet werden, d.h. zu jedem Zeitpunkt darf nicht mehr als ein SCO aktiv sein. Die Aktivierung von SCOs durch andere SCOs ist verboten. SCOs dürfen nur vom LMS gestartet werden. Beim Startvorgang ist das SCO dafür verantwortlich, den API Adapter des LMS zu lokalisieren. Das LMS stellt seinen API Adapter in einem Browserfenster in Form eines XML Document Object Model (DOM) Objekts zur Verfügung.

4.5.2.3.2 Application Programming Interface (API)

Durch die Verwendung einer **einheitlichen, standardisierten Schnittstelle** (API) können viele der Anforderungen nach Interoperabilität und Wiederverwendbarkeit erfüllt werden. Innerhalb der Schnittstelle werden Methoden definiert, die von beiden Seiten

garantiert zur Verfügung gestellt werden. Jede der Komponenten kann sich folglich darauf verlassen, dass die jeweils andere Komponente die in der Schnittstelle vereinbarten Funktionen und Methoden implementiert. Eine Kenntnis der Implementierung selbst ist dabei nicht erforderlich.

Das SCORM API wurde auf Grundlage der Guidelines for Interoperability von AICC entwickelt [AICC, 2001]. Kommunikation zwischen SCO und LMS wird immer **vom Wissensatom**, dem SCO, **initialisiert**. Folgende Kategorien von Funktionalität werden zur Verfügung gestellt [Dodds, 2001c]:

- **Execution State**: die beiden Funktionen `LMSInitialize(„")` und `LMSFinish(„")` werden vom Wissensatom benötigt, um dem LMS den **momentanen Abarbeitungsstatus** mitteilen zu können.
- **State Management**: mit Hilfe der drei Funktionen `LMSGetLastError(„")`, `LMSGetErrorString(errornumber)` und `LMSGetDiagnostic(parameter)` können **Fehlerbehandlungsroutinen** durchgeführt werden.
- **Data Transfer**: weitere drei API-Funktionen, nämlich `LMSGetValue(data model element)`, `LMSSetValue(data model element)` und `LMSCommit(„")`, werden für den **Austausch von Daten** zwischen dem LMS und dem Wissensatom benötigt.

Der SCORM API Adapter durchläuft während der Ausführung eines SCO gewisse **definierte Zustände**. Die Zustände des Adapters definieren die jeweils zum entsprechenden Zeitpunkt gültigen Aufrufe von Funktionen. Eine Übersicht der Zustände und der entsprechend gültigen Funktionsaufrufe zeigt Abbildung 43.

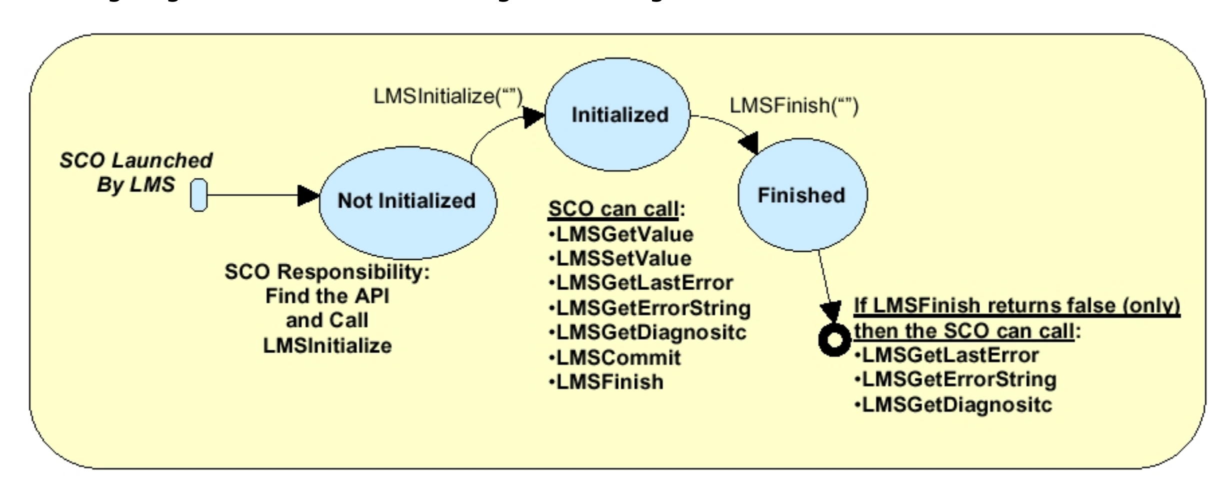


Abbildung 43: Zustandsdiagramm des SCORM API Adapters (Quelle: [Dodds, 2001c], S. „3-13")

Auf Details des SCORM wird an dieser Stelle nicht näher eingegangen. Diese sind in [Dodds, 2001c], Seite „3-7" ff. detailliert spezifiziert.

4.5.2.3.3 Data Model (Datenmodell)

Der Zweck eines einheitlichen, standardisierten Datenmodells ist es, sicherzustellen, dass verschiedene Wissenstransfer-Umgebungen **Zugriff auf wohldefinierte Daten** über

Wissensatome erhalten können. Mit anderen Worten stellt ein Datenmodell zumindest ein Vokabular zur Verfügung, mit dessen Hilfe die auszutauschenden Daten formuliert werden können. Auch dies dient wiederum dem Ziel, Interoperabilität zwischen Lernumgebungen zu realisieren.

SCORM verwendet das von der AICC entwickelte **AICC CMI Data Model**, da dieses bereits in der Praxis implementiert wurde. Es werden von den Autoren Richtlinien angeführt, die bei der Verwendung des Datenmodells Beachtung finden sollten [Dodds, 2001c]:

- Das **erste Symbol im Namen** des Datenelements dient zur Identifizierung des Datenmodells. So beginnen z.B. alle Datenelemente des AICC CMI Datenmodells mit dem Symbol „cmi". Das ermöglicht es, das selbe API auch zusammen mit anderen Datenmodellen zu verwenden und erweitert so die Funktionalität des API.
- Es gibt **drei reservierte Schlüsselwörter**, die alle mit einem Unterstrich (underscore: _) beginnen und ausschließlich mit Kleinbuchstaben benannt sind: **_version** gibt die aktuelle Versionsnummer des Datenmodells an; **_children** gibt Information über jene Datenelemente, die vom LMS unterstützt werden; **_count** wird verwendet, um die Länge von Elementlisten anzugeben.
- **Arrays** werden immer beginnend mit der Zahl 0 durchnummeriert. Objekte sollten gemäß der Indexnummern des Arrays von unten nach oben abgelegt werden, d.h. das Array sollte immer von unten beginnend befüllt werden.
- **Groß- und Kleinschreibung** der Datenelemente ist relevant.
- Jedes SCO implementiert einen **eigenständigen Satz von Datenelementen** des Datenmodells. Mit anderen Worten: ein SCO darf nicht auf die Datenelemente eines anderen SCOs zugreifen.

Weitere Details zum CMI Data Model werden an dieser Stelle nicht behandelt. Für eine Spezifikation und ausführliche Beschreibung, insbesondere auch der verwendeten Datentypen und zulässigen Werte, wird [AICC, 2001] empfohlen. Auch in [Dodds, 2001c] findet sich eine detaillierte Erläuterung.

4.6 Zusammenfassung

Im Kapitel 4 wurden mit den Problemstellungen dieser Diplomarbeit inhaltlich verwandte Konzepte, Technologien, Projekte, usw. analysiert. Gegenstand der Literatur- und Konzeptanalyse waren Konzepte zur Zerlegung von Dokumenten, Konzepte zur Speicherung zerlegter Dokumente, Konzepte zur Darstellung und Verwendung zerlegter Dokumente und schließlich Referenzprojekte und Rahmenmodelle.

Mit der Literatur- und Konzeptanalyse wurden folgende Ziele der Diplomarbeit erreicht:

- Der *state of the art* für die zur Beantwortung der Fragestellungen der Diplomarbeit bedeutenden Fachgebiete wurde geklärt und zusammengefasst.
- Aufgrund der in Literatur und in Produkten auf den Märkten enthaltenen Konzepte konnte die Grundlage für ein Datenmodell erarbeitet werden, das einen minimalen Bedarf an Speicherplatz und minimale Zugriffszeiten gewährleistet.

- Häufig verwendete und nützliche Schnittstellen für Werkzeuge wurden analysiert. Die in Referenzprojekten implementierten Werkzeuge zeigten den Bedarf für notwendige und sinnvolle Features auf, die von Werkzeugen zum Zugriff auf einen Contentpool zu erfüllen sind.

Folgende Fragestellungen konnten mit Hilfe der Konzeptanalyse ganz oder teilweise beantwortet werden:

- Die Anforderungen, die bei der Zerlegung von Dokumenten zu beachten sind, konnten geklärt werden.
- Es wurde ersichtlich, dass die semantischen Zusammenhänge zwischen Wissenseinheiten am effizientesten mit einem semantischen Netz zu modellieren sind.
- Für die Speicherung der Wissensatome in einer Datenbasis kann eine relationale Datenbank verwendet werden.
- Konzepte zur Darstellung und Verwendung von Wissensatomen konnten nur in Ansätzen gefunden werden. Dies zeigt auf, dass für diese Fragestellungen eigene Lösungen zu erarbeiten sind.

5 Bewertung der Konzepte und Technologien

Die **Relevanz und Brauchbarkeit** für die Erstellung der Architektur des Contentpools der in Kapitel 4 analysierten Konzepte (siehe Seite 35 ff.) wird im vorliegenden Kapitel untersucht. Dies geschieht mit einer Bewertung (vergleichenden Analyse), bei der das Ziel im Mittelpunkt steht, die Qualität der analysierten Konzepte in Bezug auf die Fragestellungen der Diplomarbeit zu beurteilen.

Zu diesem Zweck wird zuerst ein Bewertungssystem entworfen (Abschnitt 5.1). Danach wird die Vorgehensweise bei der Bewertung vorgestellt (Abschnitt 5.2). Die Vorgehensweise geht sowohl auf die Ziele der Bewertung, als auch auf die der Bewertung zu Grunde gelegte Methodik ein. Den Abschluss des Kapitels bildet die Präsentation der Ergebnisse der Bewertung (Abschnitt 5.3).

Es werden Merkmale definiert, die zur Bestimmung von optimal für die Beantwortung von Fragestellungen der Diplomarbeit geeigneten Konzepte dienen. Die optimalen Konzepte auf dem aktuellen Stand des Wissens in relevanten Fachgebieten werden auf Grundlage der Merkmale ausgewählt.

Mit der Durchführung der Bewertung werden folgende Ziele der Diplomarbeit verfolgt:

- Der Entwurf eines Datenmodells, das dem aktuellen Stand der Erkenntnis entspricht, wird ermöglicht.
- Es werden Technologien bestimmt, die bei der Implementierung von Werkzeugen für die Erzeugung von Wissensatomen bzw. den Zugriff auf Wissensatome zum Einsatz kommen.

5.1 Entwurf eines Bewertungssystems

In diesem Abschnitt werden auf Grundlage der in Kapitel 2 vorgestellten Designziele (vgl. dazu insbesondere Abschnitt 2.3 Ziele für das Design der Software, Seite 18 ff.), die für den Entwurf des Contentpools und dessen Werkzeuge festgelegt wurden, zwei Bewertungsraster entworfen. Es werden jeweils zahlreiche Merkmale definiert und in Kategorien gegliedert.

Die Merkmale sowie die Merkmals-Kategorien des Bewertungsrasters 1 wurden aus den Fragestellungen der Diplomarbeit abgeleitet wie nachfolgend beschrieben:

- Ob die automatisierte Zerlegung in einem Konzept enthalten ist, bzw. wie weit diese ein Konzept diese unterstützt, wird mit der Merkmals-Kategorie 1 (Erzeugung von Wissenseinheiten) bewertet.
- Die Datenmodelle, welche zur Speicherung von Wissenseinheiten dienen, werden mit der Merkmals-Kategorie 2 (Datenmodell zur Speicherung von Wissenseinheiten) bewertet. Die Merkmale in dieser Kategorie dienen zur Überprüfung, wie gut mit einem Konzept die Ziele erfüllt werden können, die die Berücksichtigung von Semantik und der effizienten Speicherung der Wissenseinheiten vorsehen.

Die Merkmale sowie die Merkmals-Kategorien des Bewertungsrasters 2 wurden primär aus folgenden beiden Fragestellungen der Diplomarbeit abgeleitet: Darstellung von Wis-

sensatomen und Verwendung der Wissensatome. Die definierten Merkmale dienen darüber hinaus zur Überprüfung, wie weit einige der Ziele der Diplomarbeit erfüllt werden können, wie nachfolgend beschrieben wird:

- Die Merkmals-Kategorien 1 (Funktionalität zur Suche in den Wissenseinheiten) und 2 (Funktionalität zur Pflege der Wissenseinheiten) stellen die Ziel in den Vordergrund, Werkzeuge für den effizienten Zugriff auf den Contentpool zu implementieren, die Effizienz des Wissenstransfers zu erhöhen und die Wiederverwendbarkeit von Lernmaterialien zu verbessern.
- Die Merkmals-Kategorien 3 (Funktionalität zum Erstellen von Dokumenten) und 4 (Funktionalität zur persönlichen Anpassung von Dokumenten) prüfen, wie weit die Ziele erreicht werden können, die Lernenden und Lehrenden in Wissenstransfer-Umgebungen bei ihren jeweiligen Aufgaben zu unterstützen.

Mit Hilfe der zwei Bewertungsraster sind die in Kapitel 4 untersuchten Konzepte, Ansätze, Projekte, Methoden und Werkzeuge zu prüfen (bewerten), inwieweit sie zur **Erfüllung der Designziele** beitragen können. Auf diese Weise kann abgeschätzt werden, welche bereits existierenden Konzepte als Grundlage für den Entwurf des Contentpools brauchbar sind. Wie bei den Zielen der Diplomarbeit angesprochen (vgl. Abschnitt 1.1, Seite 2 ff.), soll nicht „das Rad zwei Mal erfunden werden".

Die Bewertung liefert die Entscheidungsgrundlage, um festzustellen, in welchen Bereichen verwertbare Konzepte entwickelt wurden bzw. wo Lücken bestehen und deshalb eigene Konzepte zu entwickeln sind.

5.1.1 Bewertungsraster 1: Datenerzeugung (Datenspeicherung)

Im ersten Bewertungsraster stehen Aspekte im Vordergrund, die die **Erzeugung von Wissensatomen** (= Wissenseinheiten; evtl. aus Quelldokumenten) sowie die physische Speicherung der Daten betreffen. Entsprechend dieser Einteilung sind auch die Merkmale des Bewertungsraster 1 in zwei Kategorien aufgeteilt:

- Merkmale bezüglich der **Werkzeuge** (Tools), die zur Erzeugung einer Daten- oder Wissensbasis benötigt werden.
- Merkmale bezüglich des **Datenmodells** zur Speicherung der erzeugten und mit Metadaten aufbereiteten Wissenseinheiten.

Tabelle 3: Bewertungsmerkmale für Konzepte bezüglich der Speicherung von Daten

Nr.	Merkmale	Bewertung	Anmerkung
1	**Erzeugung von Wissenseinheiten**	✓ / ☒	
1.1	Automatisierte Zerlegung von Dokumenten	✓ / ☒	
1.1.1	Erkennen von Grenzen zwischen Wissenseinheiten	✓ / ☒	
1.1.2	Zuweisung von Metadaten zu den Wissenseinheiten	✓ / ☒	

1.1.3	Auffinden von semantischen Beziehungen zwischen den Wissenseinheiten	✓ / ☒	
1.1.4	Möglichkeit zur Konfiguration von Parametern nach den Präferenzen des Benutzers	✓ / ☒	
1.2	Manuelle Aufbereitung von Wissenseinheiten (Inhalte in der Daten- oder Wissensbasis)	✓ / ☒	
1.2.1	Bearbeiten der Zerlegung des Dokuments in Wissenseinheiten	✓ / ☒	
1.2.2	Bearbeiten und Ergänzen der automatisch generierten Metadaten	✓ / ☒	
1.2.3	Bearbeiten und Ergänzen der automatisch erzeugten semantischen Verknüpfungen	✓ / ☒	
2	**Datenmodell zur Speicherung der Wissenseinheiten**	✓ / ☒	
2.1	Sicherstellung eines konsistenten (einheitlichen) Begriffssystems	✓ / ☒	
2.1.1	Verwendung einer oder mehrerer Schlagwort-Tabelle(n)	✓ / ☒	
2.1.2	Verwendung eines Thesaurus bzw. mehrerer Thesauri	✓ / ☒	
2.1.3	Unterstützung mehrerer Sprachen	✓ / ☒	
2.2	Physische Speicherung von Daten	✓ / ☒	
2.2.1	Möglichkeit zur Abbildung semantischer Beziehungen zwischen den Wissenseinheiten	✓ / ☒	
2.2.2	Separate Speicherung von Wissens- und Metadaten	✓ / ☒	
2.2.3	Separate (und private) Speicherung von persönlichen Inhalten eines Benutzers	✓ / ☒	
2.2.4	Möglichkeit zur Abbildung einer Kategorisierung der Wissenseinheiten	✓ / ☒	
2.2.5	Unterstützung von Multicodalität	✓ / ☒	
2.2.6	Möglichkeit zur Speicherung von Annotationen und sonstigen personalisierten Inhalten	✓ / ☒	
2.2.7	Unterstützung technischer Standards und Spezifikationen (z.B. XML)	✓ / ☒	

2.3	Speicherung von Metadaten	✓ / ☒	
2.3.1	Strukturinformation (Autor, Buchtitel, Kapitel-Nr., Seite, laufende Nummer innerhalb der Seite, ...)	✓ / ☒	
2.3.2	Versionsnummerierung	✓ / ☒	
2.3.3	Meta-Metadaten (Information über die Metadaten, z.B. nach welcher Spezifikation sie erstellt wurden)	✓ / ☒	
2.3.4	Technische Information	✓ / ☒	
2.3.5	Pädagogische Anmerkungen	✓ / ☒	
2.3.6	Urheberrechtliche Aspekte und Einschränkungen	✓ / ☒	
2.3.7	Referenzen zwischen Wissenseinheiten (semantische Relationen)	✓ / ☒	
2.3.8	Beschlagwortung	✓ / ☒	
2.3.9	Anmerkungen des Autors der Wissenseinheit	✓ / ☒	
2.3.10	Kategorisierung der Wissenseinheit (Klassifizierung des semantischen Inhalts)	✓ / ☒	

Hinweise zur Verwendung der Tabelle 3:

- Für die Spalte „Bewertung" ist das Symbol ✓ einzutragen, falls das Konzept das betreffende Merkmal unterstützt; andernfalls ist das Symbol ☒ einzutragen.
- In der Spalte „Anmerkung" wird in Stichworten die Entscheidung begründet, die zur Vergabe der Bewertung führte.

5.1.2 Bewertungsraster 2: Tools für den Zugriff auf die Daten

Im zweiten Bewertungsraster stehen Aspekte im Vordergrund, die die Werkzeuge (Tools) für den Zugriff auf eine **bereits vorhandene** (d.h. zuvor erstellte) Daten- oder Wissensbasis beurteilen helfen. Entsprechend der Funktionalität der einzelnen Tools werden die Merkmalein folgende Kategorien aufgeteilt:

- Merkmale für Tools zur **Suche** in den Wissenseinheiten
- Merkmale für Tools zur **Pflege der Wissenseinheiten** (= Tools zur Verwaltung, das bedeutet Hinzufügen, Bearbeiten und Löschen von Wissenseinheiten)
- Merkmale für Tools zum **Erstellen von Dokumenten** aus den Wissenseinheiten
- Merkmale für Tools zur **persönlichen Anpassung** von Dokumenten

Tabelle 4: Bewertungsmerkmale für Konzepte bezüglich Tools für den Datenzugriff

Nr.	Merkmale	Bewertung	Anmerkung
1	**Funktionalität zur Suche in den Wissenseinheiten**	✓ / ☒	
1.1	Schnittstelle zur Suche innerhalb des Contentpools nach unterschiedlichen Bedingungen	✓ / ☒	
1.2	Möglichkeit zur Verknüpfung beliebig vieler Suchbedingungen mit Hilfe von boole'schen Operatoren	✓ / ☒	
1.3	Möglichkeit zur Suche nach semantischen Bedingungen (d.h. Suche nach verwandten Themen, aufbauenden Themen, Voraussetzungen usw.)	✓ / ☒	
2	**Funktionalität zur Pflege der Wissenseinheiten**	✓ / ☒	
2.1	Schnittstelle zum Updaten von Wissenseinheiten	✓ / ☒	
2.2	Schnittstelle zum Löschen von Wissenseinheiten	✓ / ☒	
2.3	Schnittstelle zum manuellen Einfügen neuer Wissenseinheiten	✓ / ☒	
2.4	Möglichkeiten zur Versionsnummerierung von Wissenseinheiten	✓ / ☒	
3	**Funktionalität zum Erstellen von Dokumenten (Kursmaterialien und Nachschlagewerke)[29]**	✓ / ☒	
3.1	Möglichkeit zur Nutzung der Suchfunktionalität	✓ / ☒	
3.2	Schnittstelle zur Erstellung eines Kursmaterials	✓ / ☒	
3.3	Schnittstelle zur Erstellung eines Nachschlagewerks	✓ / ☒	
3.4	Schnittstelle zum Wiederherstellen der Originaldokumente in der ursprünglichen Reihenfolge	✓ / ☒	
3.5	Konvertierung von Dokumenten in unterschiedliche Codalität	✓ / ☒	
4	**Funktionalität zur persönlichen Anpassung von Dokumenten**	✓ / ☒	
4.1	Schnittstelle zur Speicherung persönlicher Inhalte (eigene Bild-, Ton- und Videodokumente, eigene Texte,	✓ / ☒	

[29] Anmerkung: zur begrifflichen Unterscheidung zwischen "Kursmaterial" und "Nachschlagewerk" vgl. Kapitel 10.2 (Glossar) im Anhang, Seite 333 ff.

	usw.)		
4.2	Schnittstelle zur Erstellung und Speicherung von Annotationen	✓ / ☒	
4.3	Schnittstelle zur Erstellung und Speicherung von benutzerdefinierten Formatierungen bestimmter Textstellen	✓ / ☒	
4.4	Schnittstelle zur Auswahl einer gewünschten Codalität des Dokuments	✓ / ☒	
4,5	Bereitstellung von Exportfiltern zur Speicherung eines Dokuments in einem von der Anwendung unabhängigen Format (z.B. pdf)	✓ / ☒	

Hinweise zur Verwendung der Tabelle 4:

- Für die Spalte „Bewertung" ist das Symbol ✓ einzutragen, falls das Konzept das betreffende Merkmal unterstützt; andernfalls ist das Symbol ☒ einzutragen.
- In der Spalte „Anmerkung" wird in Stichworten die Entscheidung begründet, die zur Vergabe der Bewertung führte.

5.1.3 Vorgehensweise bei der Bewertung

Für die Bewertung (vergleichende Analyse) der in Kapitel 4 (Related Work) vorgestellten Konzepte sind die Bewertungsraster 1 und 2 (in Tabelle 3 und Tabelle 4 angeführt) anzuwenden. Dabei ist folgende Vorgehensweise anzuwenden:

- Für jedes der Konzepte ist in einem ersten Schritt zu bestimmen, ob es sich lediglich um ein Konzept zur Beschreibung (z.B. ein Modell) handelt und/oder Tools zur Bearbeitung von Wisseneinheiten, semantischen Netzen oder Datenbasen behandelt werden.
- Wenn sich ein Konzept mit der **Beschreibung eines Datenmodells** beschäftigt, ist der Bewertungsraster 1 (siehe Abschnitt 5.1.1 und Tabelle 3) zu verwenden.
- Bei Konzepten, die **Werkzeuge** (Tools) für die Erstellungs- und/oder Wartungszwecke zur Verfügung stellen, ist der Bewertungsraster 2 (siehe Abschnitt 5.1.2 und Tabelle 4) anzuwenden.
- Aus der gesamten Menge der ausgefüllten Bewertungsraster 1 und Bewertungsraster 2 wird schließlich ersichtlich, für welchen Anforderungsbereiche **brauchbare wissenschaftliche und/oder praktische Konzepte** existieren und in welchen Anforderungsbereichen eigene Konzepte entwickelt werden müssen. Als Ergebnis entsteht somit ein **Anforderungskatalog** für die Implementierung des Contentpools.

5.1.4 Metriken zur Bewertung

Die Messung des Ausmaßes der Erfüllung bezüglich der zur Bewertung (vergleichenden Analyse) definierten Merkmale ist im vorliegenden Fall alles andere als trivial. Dement-

sprechend gestaltete sich die Bewertung der gefundenen Konzepte schwierig. Messen ist „eine Zuordnung von Zahlen zu Objekten oder Ereignissen, sofern diese Zuordnung eine homomorphe Abbildung eines empirischen Relativs in ein numerisches Relativ ist" (nach [Bortz, 1993], S. 19). Es ist aus der Formulierung der Bewertungsraster (vgl. Tabelle 3 und Tabelle 4) leicht ersichtlich, dass die **numerische Bewertung** von Eigenschaften der untersuchten Konzepte nicht leicht durchführbar ist.

Die Anwendung kardinaler oder ordinaler Skalen zur Messung der Merkmalserfüllung wäre wünschenswert, ist aber, wie oben dargestellt, nicht möglich. Aus diesem Grund wurde eine **nominale Skala** gewählt, auf der lediglich die beiden Werte „**wird unterstützt**" oder „**wird nicht unterstützt**" einzutragen sind. Um diese spärliche formale Messung zu ergänzen, wird (vor allem im Zusammenhang mit dem Bewertungsraster 2 – vgl. Tabelle 4) eine zusätzliche qualitative (inhaltliche) Bewertung in Form von freien Anmerkungen durchgeführt. Letztendlich lag es im Ermessen des Autors, aufgrund der inhaltlichen Bewertung die am besten geeigneten Konzepte zu ermitteln.

5.2 Vorgehensweise für die Bewertung

Bevor die Ergebnisse der Bewertung (vergleichenden Analyse) vorgestellt werden, gehe ich auf deren Rahmenbedingungen ein. Zu diesem Zweck werden die Ziele vorgestellt, die bei der Bewertung verfolgt werden (vgl. Abschnitt 5.2.1). Weiters wird die methodische Durchführung besprochen (vgl. Abschnitt 5.2.2).

Hinweis: Die Liste der bewerteten Konzepte ergibt sich aus den in Kapitel 4 präsentierten Konzepten.

5.2.1 Ziele der Bewertung

Ziele der Bewertung sind:

- Finden eines Standards oder einer Spezifikation, der bwz. die zur **Speicherung** von Wissensatomen und Lernmaterialien geeignet ist.
- Finden eines Standards oder einer Spezifikation, der bzw. die zur **Darstellung** von Wissensatomen und Lernmaterialien geeignet ist.
- Das Identifizieren von gängigen **Anforderungen an Tools**, die für die Erstellung und Wartung einer Daten- oder Wissensbasis in Wissenschaft und Praxis verwendet werden.
- Im optimalen Fall können sogar Tools ausfindig gemacht werden, die für den Contentpool **unverändert übernommen** und verwendet werden können. Für solche Tools ist bloß die ausreichende Kenntnis der Schnittstelle von Bedeutung.

5.2.2 Methodische Durchführung

Folgende **methodische Vorgehensweise** wurde für die Durchführung der Bewertung gewählt:

- Definition von K.O.-Kriterien (vgl. Abschnitt 5.3.1, Seite 130 f.)

- Ausscheiden von ungeeigneten Konzepten an Hand der K.O.-Kriterien (vgl. Abschnitt 5.3.1)
- Bestimmung der Bewertungsmerkmale für die verbleibenden Konzepte (vgl. Abschnitt 5.1)
- Bewertung (vergleichende Analyse) der verbleibenden Konzepte an Hand der Bewertungsmerkmale (vgl. Abschnitt 5.3.2, Seite 133 ff.)
- Interpretation der Bewertungsergebnisse; Konklusion und Auswahl des oder der am besten geeigneten Konzepte(s) (vgl. Abschnitt 5.3.3, Seite 141 ff.)

Als Entscheidungsgrundlage für die Bewertung der Konzepte (d.h. für das Ermitteln von Erfüllungswerten pro Merkmal und Konzept) wurden die Ergebnisse der Literatur- und Konzeptanalyse verwendet (siehe Kapitel 4, Seite 35 ff.).

5.3 Ergebnisse der Bewertung

Es gibt bis jetzt lediglich einen einzigen offiziellen Standard für Metadaten zu Dokumenten (z.B. elektronische Lernmaterialien), nämlich LOM (Learning Object Metadata) des IEEE Learning Technology Standards Committee (LTSC). Von den bedeutendsten der in dieser Diplomarbeit analysierten Konzepte wird der Standard unterstützt. Für die Speicherung von (evtl. zerlegten) Dokumenten und semantischen Zusammenhängen zwischen Dokumenten und Teilen von Dokumenten gibt es ebenfalls einen einzigen offiziellen Standard, die XML Topic Maps.

Der **LOM-Standard** definiert die Struktur der Metadaten von *Learning Objects*. Im **XML Topic Map Standard** wird ein Format für die Modellierung semantischer Netze definiert. Die restlichen analysierten Konzepte sind Spezifikationen. Zur Unterscheidung der beiden Begriffe „Standard" und „Spezifikation" wird auf den Abschnitt 4.1.1 im Kapitel „Related Work" (Seite 36 ff.) verwiesen.

Im vorliegenden Abschnitt werden zuerst uninteressante Konzepte ausgeschieden (5.3.1 Vorauswahl aufgrund von K.O.-Kriterien). Anschließend werden die verbliebenen Konzepte detailliert bewertet (5.3.2 Bewertung der verbleibenden Konzepte) und daraus Schlussfolgerungen gezogen (5.3.3 Konklusion und Schlussfolgerung).

5.3.1 Vorauswahl aufgrund von K.O.-Kriterien

Um den **hohen Zeitaufwand** für die Durchführung der Bewertung zu reduzieren, wurden bereits vor Beginn der in Kapitel 4 dokumentierten Literatur- und Konzeptanalyse die Bemühungen auf die erfolgversprechendsten Standards und Spezifikationen konzentriert. Gleich zu Beginn der Bewertung war folglich das Ziel, ungeeignete Konzepte zu verwerfen. Dies erfolgte an Hand der hier präsentierten K.O.-Kriterien.

Die Ziele für die Erstellung des Contentpools wurden im Kapitel 2.3 Ziele für das Design der Software (Seite 18 ff.) vorgestellt. Aus diesen Zielen wurden die folgenden **K.O.-Kriterien** abgeleitet.

1. Ist das Konzept als **Open-Source Projekt**, d.h. zusammen mit dem Quellcode, zumindest aber kostenlos erhältlich?

2. Wird das Konzept von großen, bedeutenden Unternehmen unterstützt bzw. bestehen **Partnerschaften**?
3. Wird das Konzept **weltweit** oder zumindest in mehreren bedeutenden Organisationen eingesetzt? Mit anderen Worten: Hat sich das Konzept bereits bewähren bzw. durchsetzen können?
4. Werden **offene Schnittstellen** angestrebt und andere Standards und Spezifikationen unterstützt? Kann **Interoperabilität** mit anderen Konzepten bzw. Systemen erzielt werden?

Reihung (Gewichtung) der K.O.-Kriterien

Den 4 genannten K.O.-Kriterien werden unterschiedliche Prioritäten eingeräumt. Um die Reihung der Kriterien bei der Konzept-Vorauswahl einfließen zu lassen, werden entsprechend der **Priorität** für ein Kriterium **Punkte vergeben**. Tabelle 5 enthält die Übersicht der Gewichtung (in Punkten), die für die K.O.-Kriterien gewählt wurden. Je höher die Punkteanzahl für ein Kriterium, desto größer wird dessen Priorität bewertet.

Tabelle 5: Gewichtung der K.O.-Kriterien mit Punktesystem

K.O.-Kriterium	**Punkte**
Schnittstellen – Interoperabilität (4.)	5 Punkte
Open Source (1.)	3 Punkte
Einsatzhorizont (3.)	2 Punkte
Projektpartner (2.)	1 Punkt

Die Ergebnisse der Bewertung nach den K.O.-Kriterien sind in der nachstehenden Tabelle 6 angeführt. Alle im Abschnitt 3.1 (Gesamtübersicht im Kapitel 3 Auswahl der Konzepte und Technologien, Seite 27 ff.) aufgezählten Konzepte wurden auf Erfüllung der in Tabelle 5 angeführten K.O.-Kriterien geprüft.

Tabelle 6: Vorauswahl der Konzepte – K.O.-Kriterien / Konzept Tabelle

	1. **Open Source**	**2.** **Projekt-partner**	**3.** **Einsatz-horizont**	**4.** **Interoper-abilität**	**Σ** **Summe der Pkte.**
	3 Pkt.	1 Pkt.	2 Pkt.	5 Pkt.	
Slicing Book Technology	☒	✓	☒	✓	**6 Pkt.**
IEEE Learning Object Model und IEEE LOM	✓	✓	✓	✓	**11 Pkt.**

IMS Metadata & Content Packaging	✓	✓	✓	✓	**11 Pkt.**
COMpendis	☒	☒	?[30]	✓	**5 – 7 Pkt.** (*)
LMML	✓	☒	☒	☒	3 Pkt.
AICC	✓	✓	☒[31]	☒	4 Pkt.
GEM	✓	✓	☒	☒	4 Pkt.
Microsoft LRN	☒	✓	☒	☒	1 Pkt.
EML	✓	✓	☒	✓	**9 Pkt.**
XML Topic Maps	✓	✓	✓	✓	**11 Pkt.**
SIF	✓	✓	☒	☒	4 Pkt.
ARIADNE	✓	☒	☒	✓	8 Pkt.
Trial Solution	☒	✓	✓	✓	**8 Pkt.**
SCORM	✓	✓	✓	✓	**11 Pkt.**
PROMETEUS	Kein Standard/Spezifikation				n. b. (*)

(*) = Ausnahme-Regelung bei Interpretation

Hinweise zur Tabelle 6:

Für die Spalte „Bewertung" wurde das Symbol ✓ eingetragen, falls das Konzept das betreffende Kriterium unterstützt; andernfalls wurde das Symbol ☒ eingetragen. Bei Erfüllung eines Kriteriums (Symbol ✓) erhält das Konzept die dem Kriterium zugeordnete Anzahl von Punkten (vgl. Tabelle 5); bei Nichterfüllung eines Kriteriums (Symbol ☒) werden 0 Punkte vergeben. Durch Summierung der für alle Kriterien erzielten Punkte erhält jedes Konzept eine Gesamt-Punktezahl. Die Ordnung von Konzepten ergibt sich aus der aufsteigenden Reihenfolge gemäß der Gesamt-Punktezahl.

[30] Da es sich bei COMpendis um ein kommerzielles Softwareprojekt handelt (vgl. Abschnitt 4.3.4, Seite 67 ff.), sind Daten zur Verbreitung des Konzeptes schwer eruierbar.

[31] Der Fokus für den Einsatz der AICC-Spezifikationen liegt in erster Linie auf der Luftfahrt-Industrie (z.B. Flugzeug-Industrie, Pilotenausbildung, …); die Gestaltung der Spezifikationen ist zudem für diese Wirtschaftszweige optimiert. Eine Verwendung der Spezifikationen für andere Bereiche erfordert zumindest eine Anpassung, obwohl das AICC aktiv die Bildung von Standards unterstützt und mit Organisationen wie IMS oder ADL kooperiert (vgl. [AICC, 2004] bzw. http://www.aicc.org).

Interpretation der Ergebnisse

Die Ergebnisse der Vorauswahl von Konzepten auf Grund von K.O.-Kriterien gemäß Tabelle 6 wird nach folgenden Regeln interpretiert:

a. Aus jeder der 4 Kategorien für Konzepte (vgl. die Einteilung im Kapitel 3) ist zumindest ein Konzept genau zu analysieren.

b. Erreicht ein Konzept zumindest 7 Punkte, so wird es detailliert analysiert (vgl. Kapitel 4, Seite 35 ff.). Die Grenze von 7 Punkten ergibt sich aus der außerordentlichen Wichtigkeit, die dem K.O.-Kriterium „Offene Schnittstellen / Interoperabilität" zugeordnet wird. Die Punktezahl von zumindest 7 Punkten wird nur dann erreicht, wenn das genannte K.O.-Kriterium erfüllt ist. Somit wird „Offene Schnittstellen / Interoperabilität" als Muss-Kriterium angesehen.

c. Alle Konzepte, die die Grenze von 7 Punkten nicht erreichen, werden verworfen und keiner detaillierten Analyse unterzogen.

d. Im Zweifelsfall gilt Regel a. vor Regel c.

Auf Grund besonderer Erfordernisse wurden von den oben genannten Interpretationsregeln **zwei Ausnahmen** definiert:

- Beim **COMpendis Projekt** ist unklar, wie weit die entwickelte Software in Organisationen (z.B. Unternehmen) Verwendung findet. Da es sich bei COMpendis um kommerzielle Software handelt, waren Daten zum Einsatzhorizont nicht verfügbar. Da vermutlich der Einsatzhorizont ausreichend groß ist, um eine Analyse zu rechtfertigen, wurde COMpendis den zu analysierenden Konzepten zugeordnet. Die Argumentation wird von der Tatsache unterstützt, dass COMpendis das Muss-Kriterium „Offene Schnittstellen / Interoperabilität" erfüllt.

- **PROMETEUS** ist weniger ein Standard oder eine Spezifikation, als lediglich eine Plattform für den persönlichen Austausch unter Experten des E-Learning. Aus diesem Grund sind keine Ergebnisse verfügbar, die eine Literatur- oder Konzeptanalyse erlauben würden. Folglich erschien die Anwendung der K.O.-Kriterien nutzlos. Das Konzept PROMETEUS wurde daher verworfen.

Als Ergebnis der Vorauswahl auf Grund von K.O.-Kriterien werden alle Konzepte für eine detaillierte Literatur- und Konzeptanalyse heran gezogen, deren Gesamt-Punktezahl in Tabelle 6 fett gedruckt dargestellt ist. Dies entspricht all jenen Konzepten, die im Kapitel 4 untersucht wurden.

Hinweis: Eine Aufstellung all jener Konzepte, die aufgrund der in Tabelle 6 dargestellten Ergebnisse verworfen wurden, findet sich auch im Abschnitt 3.7, Seite 32.

5.3.2 Bewertung der verbleibenden Konzepte

Die Bewertung jener Konzepte, die an Hand der K.O.-Kriterien nicht verworfen und daher analysiert wurden (vgl. Tabelle 6 bzw. Kapitel 4), orientiert sich an den **zwei Bewertungsrastern**, die zu Beginn dieses Kapitels definiert wurden (vgl. Abschnitt 5.1, Seite 123 ff.). Entsprechend dieser Einteilung wird die Bewertung in folgenden zwei Teilen durchgeführt:

- Bewertung von Konzepten und Werkzeugen für die Erzeugung von zerlegten Dokumenten
- Bewertung von Konzepten für Speicherung und Zugriff auf zerlegte Dokumente

Auf den kommenden Seiten werden die Ergebnisse der Bewertung (vergleichenden Analyse) präsentiert.

5.3.2.1 Bewertung von Konzepten und Werkzeugen für die Erzeugung von zerlegten Dokumenten

Konzepte und Werkzeuge zur Erzeugung von zerlegten Dokumenten werden mit Hilfe der Merkmale bewertet, die im Bewertungsraster 1 definiert wurden (vgl. Tabelle 3 im Abschnitt 5.1.1). Für die Bewertung wurden Konzepte der Kategorie „Zerlegung von Dokumenten" heran gezogen. So weit in der Kategorie „Referenzprojekte und Rahmenmodelle" passende Ansätze enthalten waren, wurden auch Konzepte aus letzterer Kategorie heran gezogen.

Somit ergab sich die folgende **Menge an Konzepten**, die mit Bewertungsraster 1 zu bewerten waren:

- Slicing Book Technology
- SCORM

Die Ergebnisse dieser Bewertung werden in Tabelle 7 dargestellt.

Hinweis: Auch bei Trial Solution gibt es die Möglichkeit, Dokumente in zerlegter Form abzuspeichern. Das Werkzeug verwendet jedoch die Slicing Book Technology (vgl. Abschnitt 4.5.1.3, Seite 95).

Tabelle 7: Ergebnisse der Bewertung: Slicing Book Technology, SCORM

	Slicing Book Technology		**SCORM**	
Nr.	✓ / ☒	Anmerkung	✓ / ☒	Anmerkung
1	Suche in den Wisseneinheiten			
1.1	✓	Einfache Schlagwortsuche möglich.	✓	Suche in den Metadaten der Inhalte ist möglich.
1.2	☒	Nur einfache Schlagwortsuche unterstützt; Verknüpfung von Suchbedingungen nicht möglich.	☒	Aus Dokumentation ist die Möglichkeit nicht erkennbar.
1.3	☒	Suche nicht unterstützt; daher auch die Suche nach semantischen Bedingungen nicht möglich.	☒	Aus Dokumentation ist die Möglichkeit nicht erkennbar.
2	Pflege (Wartung) der Wissenseinheiten			
2.1	✓	Bearbeitung der Inhalte ist möglich.	✓	Der Inhalt von SCOs kann bearbeitet werden.

2.2	✓	Durch Wartung des Dateienbaumes möglich.	✓	
2.3	☒	Nur manuelle Nachbearbeitung der Zerlegung möglich.	✓	SCOs können beliebig erstellt, verändert oder gelöscht werden.
2.4	✓	In den Metadaten unterstützt.	✓	In den Metadaten unterstützt.
3	Erstellen von neuen Dokumenten			
3.1	✓		✓	
3.2	☒	Einbau von interaktiven Elementen (auch zur Leistungsüberprüfung) nicht möglich.	☒	Einbau von Elementen zur Leistungsüberprüfung nicht möglich.
3.3	✓	Neue Dokumente können aus dem Dateienbaum zusammen gestellt werden.	✓	Beliebige Kombination von SCOs zu neuen Dokumenten möglich.
3.4	✓	Aus den Metadaten ersichtlich.	☒	
3.5	✓	Sofern Teile der Dokumente in unterschiedlicher Codalität vorhanden sind, möglich.	☒	Codalität ist von Inhalt des SCOs abhängig.
4	Persönliche Anpassung von Dokumenten			
4.1	✓	Eigene Inhalte können gespeichert werden.	✓	Eigene Inhalte können gespeichert werden.
4.2	☒	Annotationen sind nicht möglich.	☒	Annotationen sind nicht möglich.
4.3	(✓)	Formatierungen sind möglich; können aber nicht benutzerspezifisch gespeichert werden.	☒	Erscheinungsbild hängt von der Zusammenstellung des Materials ab.
4.4	✓	Bei Vorhandensein unterschiedlicher Codalitäten kann eine ausgewählt werden.	☒	Unterschiedliche Codalitäten werden nicht unterstützt.
4.5	☒		☒	

Hinweise zu Tabelle 7:

- Für die Spalte „Bewertung“ wurde das Symbol ✓ eingetragen, falls das Konzept das betreffende Kriterium unterstützt; andernfalls wurde das Symbol ☒ eingetragen.
- In der Spalte „Anmerkung“ wird in Stichworten die Entscheidung begründet, die zur Vergabe der Bewertung führte.

5.3.2.2 Bewertung von Konzepten für Speicherung und Zugriff auf zerlegte Dokumente

Konzepte und Werkzeuge für die Speicherung von Inhalten aus zerlegten Dokumenten sowie den Zugriff auf die Inhalte werden mit Hilfe der Merkmale bewertet, die im Bewertungsraster 2 definiert wurden (vgl. Tabelle 4 im Abschnitt 5.1.2). Für die Bewertung wurden Konzepte der Kategorien „Speicherung von zerlegten Dokumenten" und „Darstellung von zerlegten Dokumenten" heran gezogen. Beide genannten Kategorien wurden, da sie anders schlecht vergleichbar wären, getrennt bewertet. So weit in der Kategorie „Referenzprojekte und Rahmenmodelle" passende Ansätze enthalten waren, wurden jeweils auch Konzepte aus letzterer Kategorie heran gezogen.

Somit ergab sich die folgende **Menge an Konzepten** aus der Kategorie „**Speicherung** von zerlegten Dokumenten", die mit Bewertungsraster 2 zu bewerten waren:

- IEEE LOM (einschließlich Learning Object Model)
- IMS Spezifikationen (IMS Metadata und IMS Content Packaging)
- COMpendis

Für die Kategorie „**Darstellung** von zerlegten Dokumenten" ergab sich folgende Menge an Konzepten, die mit Bewertungsraster 2 zu bewerten waren:

- XML Topic Maps (XTM)
- SCORM
- Educational Modelling Language (EML)

Zur Bewertung von Konzepten zur Datenhaltung wurde der in Abschnitt 5.1.2 (siehe Seite 126 ff.) vorgestellte Bewertungsraster verwendet. Auf die Merkmale des Punktes 1 wurde gegebenenfalls verzichtet, da reine Datenstrukturen nicht mit Merkmalen für Werkzeuge bewertet werden können.

Die Ergebnisse der Bewertung der oben genannten Standards werden in Tabelle 8 und Tabelle 9 präsentiert.

Tabelle 8: Ergebnisse der Bewertung: LOM, IMS, COMpendis

	IEEE LO(M)		**IMS**		**COMpendis**	
Nr.		Anmerkung		Anmerkung		Anmerkung
1	☒		☒		(✓)	
Zerlegung von Dokumenten						
1.1					☒	Zerlegung von Dokumenten wird nicht unterstützt.

Aufbereitung von Wissenseinheiten						
1.2					✓	
1.2.1					✓	Wissenseinheiten können manuell eingegeben werden.
1.2.2					✓	Metadaten werden in den Relationen gespeichert.
1.2.3					✓	Semantische Verknüpfungen können in der Relation „SEMANTISCHESNETZ" gespeichert werden.
2						
BEGRIFFSSYSTEM						
2.1	(☒)		(☒)		✓	
2.1.1	☒	Nicht vorgesehen, muss in der Anwendung implementiert werden.	☒	Nicht vorgesehen, muss in der Anwendung implementiert werden.	✓	Es wird eine Komponente „Mehrsprachiges Wörterbuch" verwendet.
2.1.2	☒	Nicht vorgesehen, muss in der Anwendung implementiert werden.	☒	Nicht vorgesehen, muss in der Anwendung implementiert werden.	✓	Es wird eine Komponente „Thesaurus" verwendet.
2.1.3	✓	Language-Feld in Metadata	✓	Feld „Language" in Metadata-Spezifikation	✓	Relation WE_WSPRACH
DATENHALTUNG						
2.2	✓		✓	Vgl. „ IMS Content Packaging Information Model"	✓	
2.2.1	☒	Ist weder im Learning Object Model noch im LOM Standard vorgesehen.	(✓)	Es kann innerhalb von Strukturen auf andere Kursstrukturen verwiesen werden (Content Packaging)	✓	Relation „SEMANTISCHESNETZ"
2.2.2	✓	Trennung zwischen Learning Objects und den Metadaten.	✓	Trennung in den Spezifikationen sichtbar	✓	Separate Speicherung in unterschiedlichen Relationen.
2.2.3	☒	Keine benutzerspezifischen Daten speicherbar.	☒	Keine benutzerspezifischen Daten speicherbar. Ausnahme: Metainformation über Name des Autors.	☒	Keine benutzerdefinierten Daten speicherbar.
2.2.4	☒	Nicht explizit vorgese-	☒		☒	

		hen.				
2.2.5	✓	Alle beliebigen Datentypen für Inhalte werden unterstützt. Sogar nicht-digitale Entitäten können als Learning Object gelten.	✓	Beliebige Datentypen für Inhalte werden unterstützt.	✓	Wird mit Komponente „multimediale Wissensrepräsentation" unterstützt.
2.2.6	☒	Nicht vorgesehen.	☒	Nicht vorgesehen.	☒	Nicht vorgesehen.
2.2.7	✓	LOM ist selbst ein Standard, bezieht aber andere Standards ein.	✓	Unterstützung von LOM.	?	Leider keine Dokumentation dazu verfügbar.
METADATEN						
2.3	✓	Vgl. IEEE LOM Standard.	✓	Vgl. „IMS Learning Resource Meta-Data Information Model"	✓	Im semantischen Netz möglich, aber nicht zu Standards konform!
2.3.1	✓		✓		✓	Nicht LOM-konform!
2.3.2	✓		✓	Versionsnummer vorhanden, Konzept offen	☒	Keine Versionierung vorgesehen.
2.3.3	✓		✓		☒	
2.3.4	✓	Format, size, location, ...	✓	Format, size, location, ...	☒	
2.3.5	✓		✓		☒	
2.3.6	✓	Rights, costs, copyright, ...	✓	Rights, costs, copyright, ...	☒	
2.3.7	✓		✓		✓	Relation „SEMANTISCHESNETZ"
2.3.8	✓		✓	Unordered list, 1000 chars	☒	
2.3.9	✓		✓		☒	
2.3.10	✓		✓		✓	Über semantisches Netz möglich.

Tabelle 9: Ergebnisse der Bewertung: XML Topic Maps, SCORM, EML

	XML Topic Maps		**SCORM**		**EML**	
Nr.		Anmerkung		Anmerkung		Anmerkung
1	☒				☒	

Zerlegung von Dokumenten						
1.1			☒	Zerlegung von Dokumenten wird nicht unterstützt.		
Aufbereitung von Wissenseinheiten						
1.2			✓			
1.2.1			✓	Zerlegung von SCOs ist beliebig editierbar.		
1.2.2			✓	Manuelle Bearbeitung der Metadaten von SCOs ist jederzeit möglich.		
1.2.3			(✓)	Ist durch Veränderung der Reihenfolge von Items eines SCOs bedingt unterstützt.		
2						
BEGRIFFSSYSTEM						
2.1	✓		☒		☒	
2.1.1	✓	Durch die Verwendung einer eigenen Topic Map können synonyme Begriffe modelliert werden.	☒	Nicht vorgesehen, muss in der Anwendung implementiert werden	☒	Nicht vorgesehen, muss in der Anwendung implementiert werden
2.1.2	✓	Thesauri können ebenfalls mit Hilfe von Topic Maps nachgebildet werden.	☒	Nicht vorgesehen, muss in der Anwendung implementiert werden	☒	Nicht vorgesehen, muss in der Anwendung implementiert werden
2.1.3	✓	Durch Topics modellierbar.	✓	Language-Feld in Metadata	☒	Kein Language-Feld vorgesehen.
DATENHALTUNG						
2.2	✓	Als Occurrences von Topics in Topic Maps.	✓	Vgl. „SCORM Content Aggregation Model"	✓	Vgl. eml10.dtd des EML 1.0 Packages
2.2.1	✓	Associations zwischen Topics.	✓	Gleiches Konzept wie IMS; Verweise innerhalb von SCOs auf andere Items.	✓	Angabe von „Prerequisite"
2.2.2	✓	Semantische Beziehungen in Topic Maps; Inhalte in Occurrences.	✓	Content Model, Metadata, Content Packaging, SCO-Metadata	☒	Metadaten sind direkt im XML-File eingebunden, keine seperate Speicherung
2.2.3	✓	In Topic Maps realisierbar.	☒	Nicht unterstützt.	☒	Nicht unterstützt.

2.2.4	✓	In Topic Maps realisierbar.	☒	Nicht unterstützt.	☒	Nicht unterstützt.
2.2.5	✓	Occurrences können Referenzen auf beliebige Dateitypen enthalten.	✓	SCOs können beliebige Datentypen enthalten.	✓	Es können unendlich viele, beliebige Objekte zu einem Content-Object hinzugefügt werden.
2.2.6	✓	Durch die Speicherung in Topic Maps ist die Speicherung von Annotationen modellierbar.	☒	Nicht unterstützt.	?	Konnte nicht bewertet werden: in den Quellen leider keine Angaben hierzu.
2.2.7	✓	XML-Standard; Einbindung anderer Standards einfach möglich.	✓	Metadata: LOM-, IMS-kompatibel.	(✓)	XML-Standard, nicht zu LOM konform.
METADATEN						
2.3	✓	Beliebige Metadaten modellierbar; auch Standards können eingebunden werden.	✓	Vgl. „SCORM Meta-data Information Model”	✓	Vgl. Metadata.dtd des EML 1.0 Packages, nicht LOM-konform!
2.3.1	✓		✓		✓	
2.3.2	✓		✓	Feld für Version	☒	Historyfeld vorgesehen, aber keine wirkliche Versionierung
2.3.3	✓		✓	Eigener Punkt	✓	
2.3.4	✓		✓	Format, size, location, ...	☒	Nur Objekt-Typ
2.3.5	✓		✓		☒	
2.3.6	✓		✓		✓	Copyright-year, -owner, -statement
2.3.7	✓		✓		☒	
2.3.8	✓		✓	LangStringType, 1000 chars	✓	Schlagwortfeld
2.3.9	✓		✓		✓	Creator, Comment-Feld
2.3.10	✓		✓		☒	

Hinweise zu Tabelle 8 und Tabelle 9:

- Für die Spalte „Bewertung“ wurde das Symbol ✓ eingetragen, falls das Konzept das betreffende Kriterium unterstützt; andernfalls wurde das Symbol ☒ eingetragen.
- In der Spalte „Anmerkung“ wird in Stichworten die Entscheidung begründet, die zur Vergabe der Bewertung führte.

5.3.3 Konklusion und Schlussfolgerung

Folgende Schlussfolgerungen können aus der Bewertung von Konzepten und Werkzeugen für die Erzeugung von zerlegten Dokumenten (vgl. Tabelle 7) gezogen werden:

- Die **Slicing Book Technology** schneidet im Vergleich zu SCORM besser ab. Es wird mehr Funktionalität für die Zerlegung von Dokumenten geboten.
- Leider sind die Ergebnisse der Slicing Book Technology **nicht offen verfügbar**. Schnittstellen der Werkzeuge wären zwar bekannt; es handelt sich jedoch bei Slicing Book Technology um ein kommerzielles Projekt. Die Nutzung der Werkzeuge wäre mit Kosten verbunden.
- Folglich ist zur Zerlegung von Dokumenten ein **eigener Algorithmus** zu entwerfen und in einem Werkzeug (oder mehreren Werkzeugen) zu implementieren.

Aus der Bewertung von Konzepten für Speicherung und Zugriff auf zerlegte Dokumente (vgl. Tabelle 8 und Tabelle 9) können folgende Schlussfolgerungen gezogen werden:

- Speicherung von zerlegten Dokumenten (Tabelle 8: Ergebnisse der Bewertung: LOM, IMS, COMpendis, Seite 136):
    - **COMpendis** bietet als einziges Konzept **auch Werkzeuge** an. Da es sich bei COMpendis aber um ein **kommerzielles Projekt** handelt, scheidet die Möglichkeit einer Verwendung der Konzepte von COMpendis aus.
    - IMS und LOM sind im Großen und Ganzen gleichwertig. Da IMS den LOM Standard als Grundlage verwendet und erweitert (vgl. [IMS Meta, 2001], [IMS Content, 2001]), orientiert sich die Architektur des Contentpools an den Spezifikationen des **IMS Konsortiums**. Dies gilt für die Spezifikationen „IMS Metadata" und „IMS Content Packaging".
- Darstellung der Inhalte aus zerlegten Dokumenten (Tabelle 9: Ergebnisse der Bewertung: XML Topic Maps, SCORM, EML, Seite 138):
    - SCORM bietet als einziges der drei Konzepte **Werkzeugunterstützung** in beschränktem Ausmaß (nur für die Bearbeitung von Wissenseinheiten) an.
    - Mit Abstand die beste Bewertung erzielt aber das Konzept der XML Topic Maps. Sie bieten größtmögliche **Flexibilität** bei der Gestaltung eines einheitlichen Begriffssystems, der Datenspeicherung und der Speicherung von Metadaten.
    - Die Architektur des Contentpools orientiert sich daher bei der Darstellung der Inhalte aus zerlegten Dokumenten am **XML Topic Maps** Standard.

5.4 Zusammenfassung

Im Kapitel 5 wurden für folgende Themenbereiche Konzepte ausgewählt, die für die Beantwortung der Fragestellungen der Diplomarbeit die optimale Unterstützung bieten:

- Für die Erzeugung von zerlegten Dokumenten existieren in Forschung und Praxis wenige Ansätze. Die Slicing Book Technology wurde als das optimale Konzept be-

stimmt. Da jedoch die Ergebnisse des Slicing-Book-Projektes nicht offen verfügbar sind, muss für die Zerlegung von Dokumenten ein eigener Algorithmus entwickelt werden.

- Die Speicherung von zerlegten Dokumenten kann auf Grundlage der Spezifikationen des Konsortiums IMS realisiert werden. Das dabei entstehende Datenmodell muss auf die Anforderungen des Contentpool-Konzepts angepasst werden. Die Speicherung von Daten in einer relationalen Datenbank ist möglich.
- Die Darstellung der Inhalte zerlegter Dokumente wird von XML Topic Maps bestmöglich unterstützt. XML Topic Maps bieten zudem flexible Möglichkeiten bei der Speicherung von semantischen Zusammenhängen in semantischen Netzen.
- Für die Verwendung von Wissensatomen sowie den Zugriff auf den Contentpool existieren keine vergleichbaren Konzepte. Es sind eigene Werkzeuge zu designen und zu entwickeln, die diese Anforderungen erfüllen.

Folgendes Ziel der Diplomarbeit wurde erfüllt:

- Konzepte als Grundlage des Datenmodells für den Contentpool wurden so gewählt, dass Interoperabilität und Offenheit gewährleistet sind. Für das Design des Contentpool-Konzepts bleibt die Aufgabe, Speicherbedarf und Zugriffszeit zu minimieren.

Folgende Ziele der Diplomarbeit wurden teilweise erreicht:

- Die Anforderungen für ein Werkzeug zur Zerlegung von Dokumenten können durch die Analyse der Slicing Books Technology klarer definiert werden.
- Durch die Bestimmung der Grundlagen für das Datenmodell des Contentpools kann die Datenschnittstelle für Werkzeuge zum Zugriff auf den Contentpool spezifiziert werden.

6 Design der Software

Im Rahmen der Analysephase des Projekts Scholion WB+ wurden grundlegende Entscheidungen zur Architektur des Contentpools getroffen. In den vorangegangenen Kapiteln der Diplomarbeit erfolgte eine Vorstellung von vorhandenen sowie die Auswahl von für die Contentpool-Verwaltung relevanten Konzepte, Ansätze, Standards und Spezifikationen (vgl. dazu die Kapitel 2: Ausgangssituation und Problemstellung, 3: Auswahl der Konzepte und Technologien, 4: Related Work, und 5: Bewertung der Konzepte und Technologien). Der Zweck des vorliegenden Kapitels ist nun, alle Entscheidungen zu dokumentieren, die für die Phase des **Systementwurfs** (Design-Phase) getroffen wurden.

Im vorliegenden Kapitel 6 wird auf Grundlage der Konzeptanalyse in Kapitel 4 und den Schlussfolgerungen aus der Bewertung von Konzepten im Kapitel 5 die Architektur des Contentpools entworfen. Besonders wichtig ist vor allem die **Festlegung der Schnittstellen** zwischen der Contentpool-Verwaltung und dem Client-Frontend. Die Design-Entscheidungen lassen sich in folgende Hauptkategorien einteilen:

- **Modellierung der Architektur** der Contentpool-Verwaltung sowie Darlegung der konzeptuellen Eingliederung derselben in das Projekt Scholion WB+ (Abschnitt 6.1 Systemarchitektur: Scholion WB+)
- Entwurf der **internen Abläufe** des Systems sowie der **Interaktion** zwischen System und Benutzer (Abschnitt 6.2 Modellierung der Prozesse im System)
- Spezifikation der benötigten **Daten** (Abschnitt 6.3 Datenmodell)
- Entwurf eines **Algorithmus** für die **Zerlegung** elektronischer Dokumente (Abschnitt 6.4 Entwurf des Splitter-Algorithmus).

Im aktuellen Kapitel werden zahlreiche Begriffe verwendet, deren Verständnis beim Leser vorausgesetzt wird. Für die Definition der Begriffe wird deshalb hier ausdrücklich auf das Kapitel 10.2: Glossar (Seite 333 ff.) verwiesen.

6.1 Systemarchitektur: Scholion WB+

In diesem Kapitel wird die Architektur des Scholion WB+ Softwaresystems entworfen. Dabei werde ich sowohl auf die Zusammenhänge zwischen dem **Contentpool** und dem **Client-Frontend** (siehe Abschnitt 6.1.1 Übersicht) als auch auf Details der **Contentpool Architektur** (siehe Abschnitt 6.1.2 Konzeptuelle Architektur des Contentpools) eingehen.

Darüber hinaus werden Aspekte der **Software- und der Verteilungsarchitektur** berücksichtigt (siehe Abschnitt 6.1.3 Software- und Verteilungsarchitektur). Dies ist insbesondere für die Definition und Darstellung der **Schnittstellen** zwischen den beiden Teilen der Wissenstransfer-Anwendung Scholion WB+ wichtig. Schließlich ist noch die Definition des **Klassenmodells** (siehe Abschnitt 6.1.4 Klassenmodell für die Contentpool-Verwaltung) Gegenstand dieses Kapitels.

6.1.1 Übersicht

Eine wichtige Design-Entscheidung betreffend die Gestaltung der Architektur des Client-Frontends ist jene zur Verwendung des **MVC Konzepts**. Dabei handelt es sich um ein Konzept für die Gestaltung von Web-Anwendungen, das eine **strikte Trennung** von Daten, Layout und Verarbeitungslogik erlaubt. Durch die Kombination eines **Java Servlets** mit einem **XSLT Prozessor** oder durch ein Framework kann diese Trennung realisiert werden (vgl. [Fürlinger, 2003]). Die zuerst genannte Möglichkeit (die Verwendung eines Servlets in Kombination mit einem XSLT Prozessor) wird in Abbildung 44 illustriert.

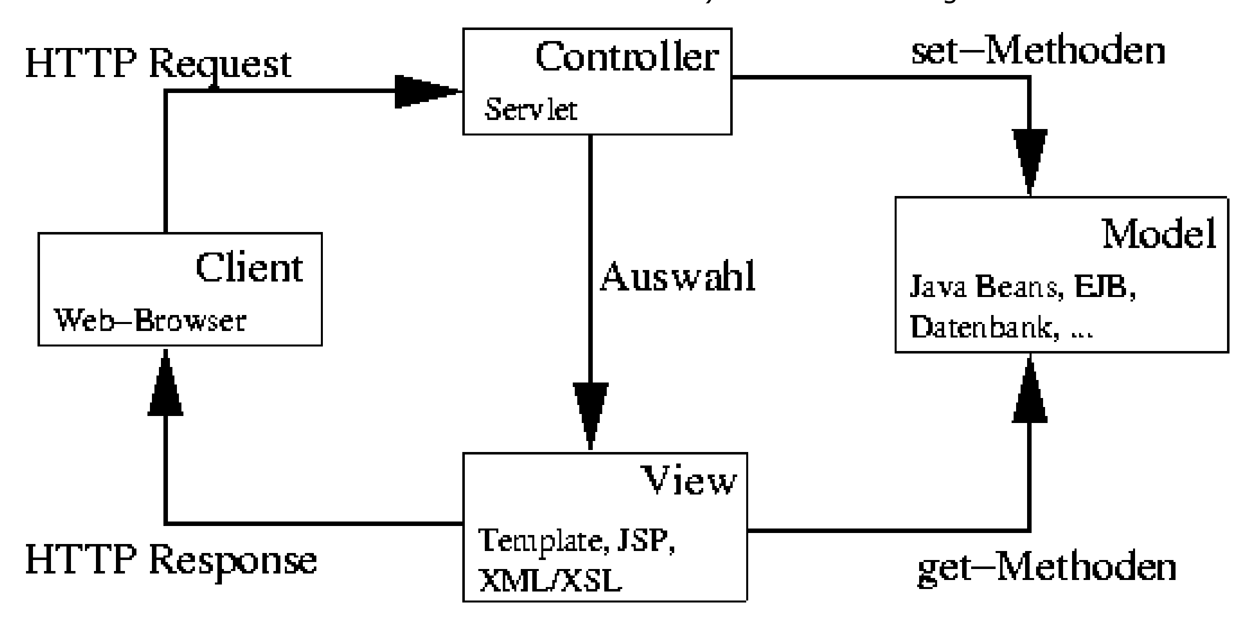


Abbildung 44: Allgemeines MVC Konzept einer Web-Anwendung in Java (Quelle: [Java, 2002])

Model-View-Controller Architekturen versprechen in Zusammenarbeit mit grafischen GUI Editoren eine hohe Produktivität und eine saubere **Trennung von Präsentation, Daten und Logik**. Die Verarbeitungslogik (*Controller*-Aufgaben) übernimmt ein **zentrales Java-Servlet**, welches alle Anfragen an die Web-Anwendung entgegen nimmt und über die auszuführende Anwendungslogik entscheidet. Die Anwendungslogik greift auf die **zugrunde liegenden Daten** (das *Model*) zurück und ändert diese. Anschließend wird eine dem jeweiligen Anwendungszustand entsprechende **Darstellung** (*View*) ausgewählt und durch ein Template, eine JSP oder XML/XSL realisiert. Die View erhält den dynamischen Anteil der Daten aus dem Model und wird anschließend an den Client geschickt (Vgl. [Fürlinger, 2003], S. 136 ff.).

Bei herkömmlichen Webanwendungen liegt der Fokus vor allem auf der Darstellung. Bei umfangreicheren und dynamischen Webinhalten müssen jedoch sowohl der Inhalt (Datenbasis), die Logik (in Form des Servlets) als auch die Präsentation und Darstellung (mit XML und XSL/XSLT) genau voneinander getrennt werden.

Wie in der Abbildung 44 dargestellt, stellen Java Server Pages aufgrund ihrer vordefinierten und relativ eng beschränkten Syntax nicht die einzige und wohl auch nicht die beste Möglichkeit für die Realisierung der View-Komponente dar. Deutliche Vorteile bezüglich

Offenheit und Flexibilität bieten **Java Servlets in Verbindung mit XML** (vgl. [Fürlinger, 2003], S. 138). Die sich dadurch ergebenden Änderungen auf das MVC Konzept sind in Abbildung 45 dargestellt.

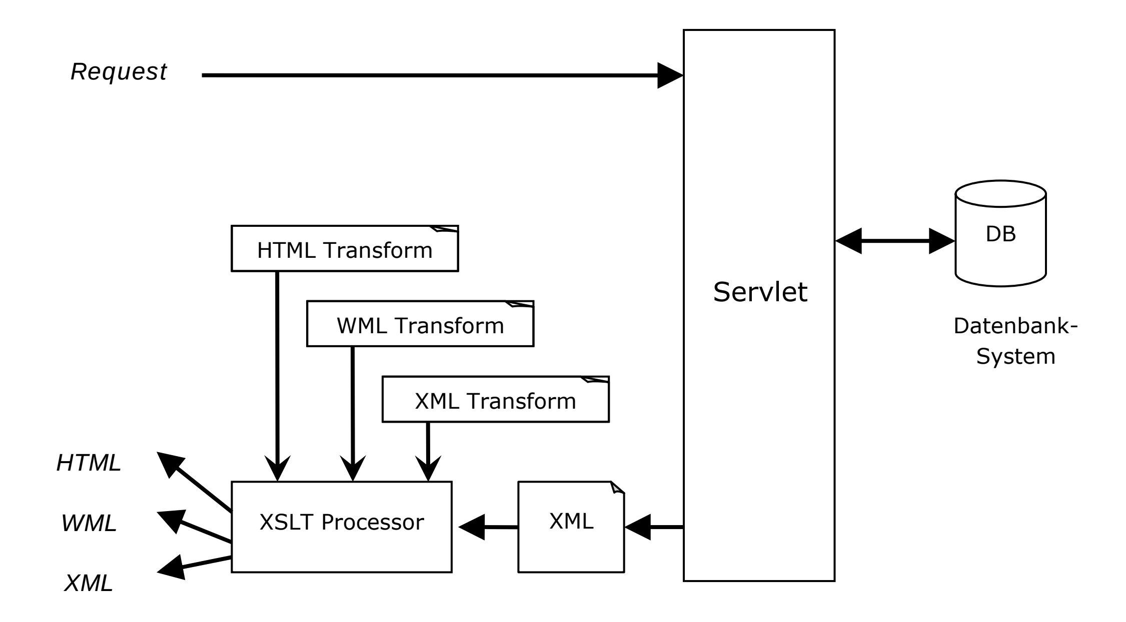


Abbildung 45: XML-spezifisches MVC-Konzept einer Web-Anwendung (Quelle: nach [Fürlinger, 2003])

Wie in der Abbildung 45 veranschaulicht, können nach der dynamischen Generierung der Inhalte die Ergebnisse in **beliebige Dokumentformate** (wie z.B. HTML oder PDF) transformiert werden, wobei die Transformation entweder beim Client oder schon beim Server vorgenommen werden kann. Moderne Webbrowser ermöglichen die sofortige Darstellung von XML in HTML-Format mit Hilfe einer DTD und eines zu XSL kompatiblen Stylesheets. Fehlt diese Möglichkeit, muss die Transformation am Server durchgeführt werden (vgl. [Fürlinger, 2003]).

Die Vorteile, die sich aus der Verwendung von XML ergeben, fasst [Fürlinger, 2003] in der folgenden Abbildung 46 zusammen.

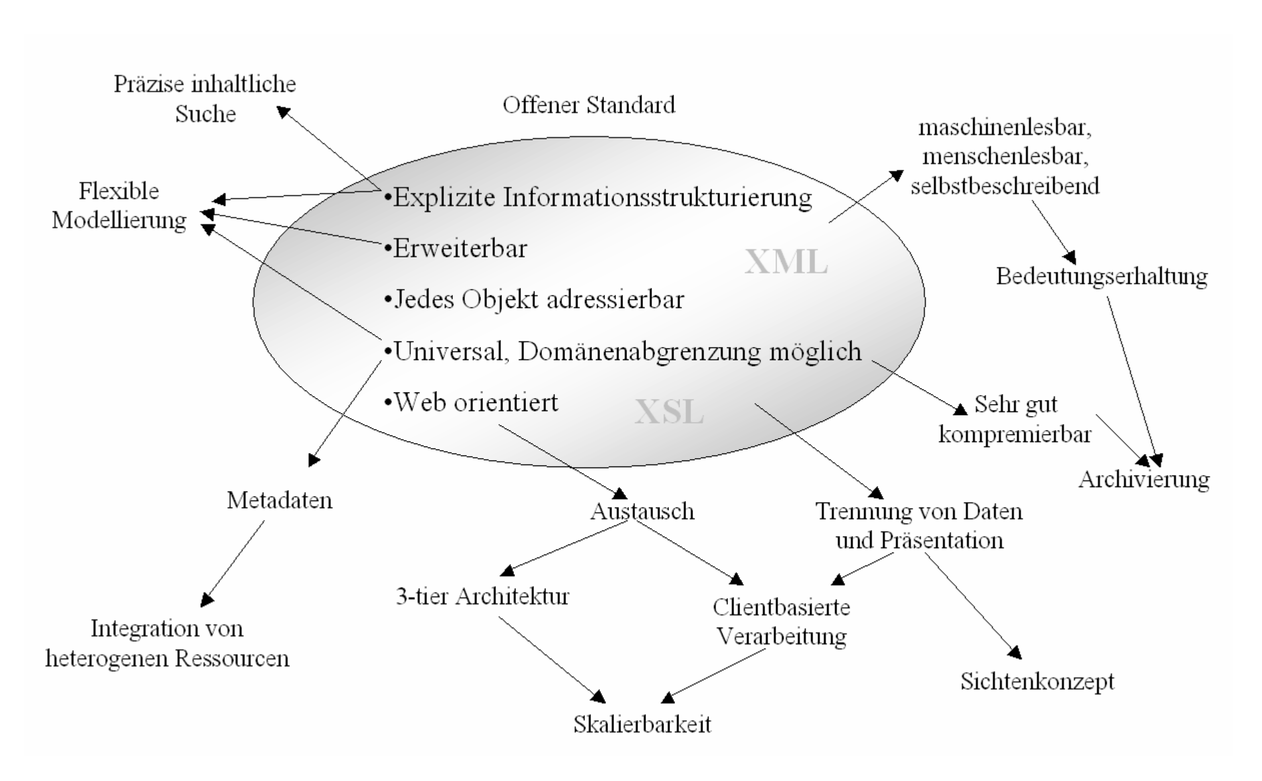


Abbildung 46: Vorteile einer auf XML basierenden Architektur (Quelle: [Fürlinger, 2003])

Da es sich bei Scholion WB+ um eine komplexe Webanwendung handelt, wird das MVC Grundkonzept verwendet und entsprechend angepasst und erweitert. Zunächst muss das Servlet in ein „Mainservlet“ und „Add-on Packages“ aufgeteilt werden. Das **Mainservlet** ist für die Kommunikation und den Datenaustausch zwischen Server und Client verantwortlich und implementiert sämtliche Grundfunktionalität, während in den **Add-on Packages** die eigentliche Funktionalität und Verarbeitungslogik implementiert ist.

Des Weiteren muss der **Contentpool** in der Architektur berücksichtigt werden. Zusätzlich sind auch Möglichkeiten zur Darstellung und Bearbeitung von im Contentpool gespeicherten Inhalten vorgesehen. Dies geschieht mit Hilfe zweier, teilweise in JavaScript implementierter Werkzeuge, die auf Grundlage des Document Object Model (DOM) arbeiten: der Topicmap-Navigator (`tm_navigate.js`) sowie der Topic-Editor (`tm_edit.js`).

Das sich aus diesen Änderungen ergebende Gesamtkonzept für die Architektur von Scholion WB+ ist nachfolgend in der Abbildung 47 dargestellt.

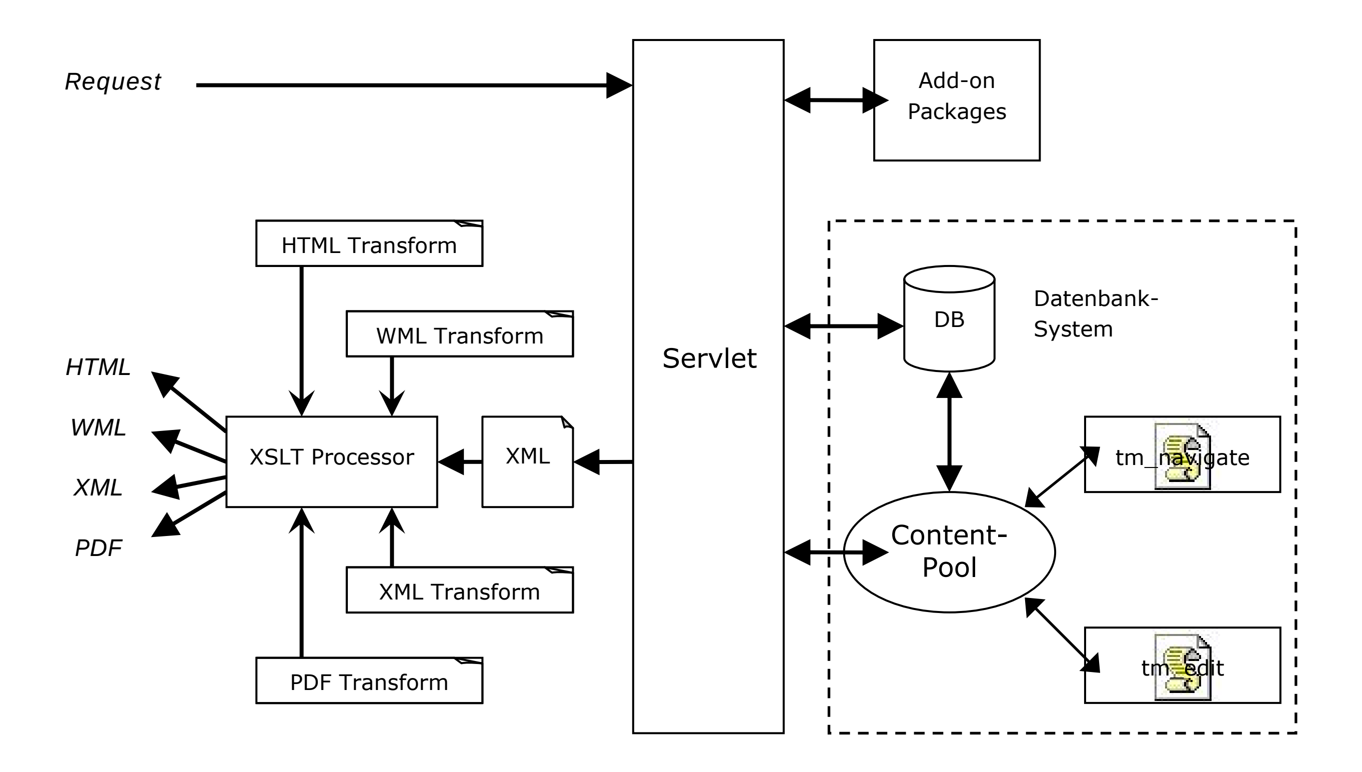


Abbildung 47: Erweitertes XML-spezifisches MVC-Konzept von Scholion WB+

Weitere Details zu den in Abbildung 47 gezeigten Komponenten des Konzepts folgen in den Abschnitten 6.1.2: Konzeptuelle Architektur des Contentpools (Seite 147 ff.) und 6.1.3: Software- und Verteilungsarchitektur (Seite 153 ff.).

6.1.2 Konzeptuelle Architektur des Contentpools

In diesem Abschnitt wird der in Abbildung 47 (Erweitertes XML-spezifisches MVC-Konzept von Scholion WB+, siehe Seite 147) durch das gestrichelte Rechteck hervorgehobene Teil detailliert dargestellt. Es handelt sich dabei um die **grundlegenden Konzepte** für die Gestaltung der Architektur jener Teile des Softwaresystems, deren Aufgabe die **Erstellung und Verwaltung von Content** in Form von Wissensatomen ist.

Wie aus der Abbildung 48 (siehe Seite 148) ersichtlich ist, wird das Scholion WB+ System in vier konzeptuelle Ebenen gegliedert.

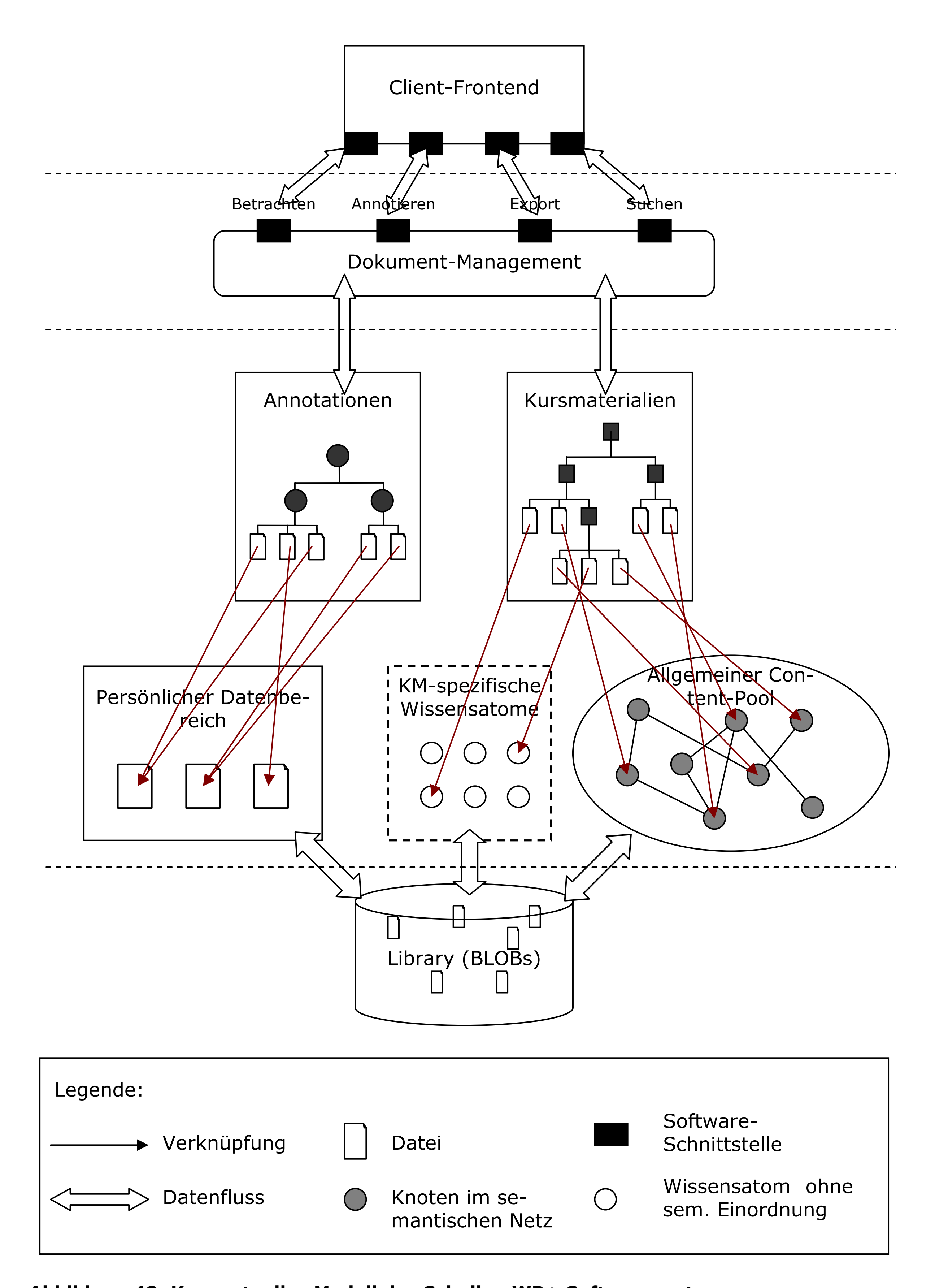


Abbildung 48: Konzeptuelles Modell des Scholion WB+ Softwaresystems

Die Komponenten des in Abbildung 48 gezeigten Konzeptes können wie folgt näher beschrieben werden.

- Im **Client-Frontend** erfolgt die **Benutzerinteraktion**. Andererseits ist diese oberste Schicht des Systems für die **Präsentation** von gemeinsamen und persönlichen Daten zuständig. Wie in der Abbildung dargestellt, wird innerhalb dieser Komponente eine Schnittstelle zum Dokument-Management angeboten, mit deren Hilfe der Benutzer Lernunterlagen betrachten, annotieren und aus den gespeicherten Inhalten ein eigenes Kursmaterial oder Nachschlagewerk zusammen stellen kann.
- Die Dokumentenverwaltung (**Dokument-Management**) ist als Zwischenschicht unterhalb der Präsentation, aber oberhalb der Datenspeicherung die **zentrale Schnittstelle** zwischen dem Client-Frontend und der Content-Verwaltung. Es werden Werkzeuge angeboten, die die Funktionalität zum Umgang mit den Lernunterlagen (betrachten, exportieren, annotieren, erstellen bzw. suchen) ermöglichen. Zudem erfolgt sämtlicher dafür notwendiger Datentransfer über die Dokument-Management Schicht.
- Der **Contentpool** besteht aus zwei Hauptteilen: dem **persönlichen Datenbereich**, in dem individuelle Wissensatome und digitale Ressourcen / Dokumente abgelegt sind, sowie dem **allgemeinen Contentpool**, der die zerlegten Bücher enthält, welche in Form von Wissensatomen allen berechtigten Benutzern des Scholion WB+ Systems zur Verfügung gestellt werden. Einen Sonderfall stellt jener Bereich dar, in welchem Wissensatome abgelegt sind, die zur Erstellung eines Kursmaterials erstellt wurden, jedoch nicht aus der Zerlegung eines Buches oder sonstigen Dokuments entstanden sind. Zusätzlich ist über die physisch gespeicherten Wissensatome ein **semantisches Netz** zum Zweck effizienter Suchzugriffe gelegt. Schließlich stellt der Contentpool auch noch **Schnittstellen** zur Erstellung von Wissensatomen und zur Verwaltung des Bestands an Wissensatomen zur Verfügung, welche mit der Dokument-Management Schicht zusammen arbeiten.
- Das **semantische Netz** erlaubt die Modellierung und Speicherung von Aussagen, die über Phänomene der realen Welt getroffen werden können. Dieses kann man sich als eine Art **erweiterten Index** für den Umgang mit dem Content vorstellen. Doch das ist nicht alles – das semantische Netz bringt den entscheidenden Nutzen im Vergleich zu herkömmlichen Büchern. Mit dessen Hilfe können **Informations-Blöcke** (so genannte Wissensatome) in **semantischen Strukturen** abgelegt werden. Unter anderem kann damit eine wesentlich erweiterte Suchfunktionalität implementiert werden. Eine erweiterte Suche ist jedoch nur unter der Voraussetzung möglich, dass umfangreiche **Metadaten** zusammen mit den Wissensatomen gespeichert werden.
- Im **persönlichen Datenbereich** ist die Speicherung von benutzerdefinierten Lernunterlagen, persönlichen Inhalten (z.B. Videodateien, Audiodateien, Dokumente, Webseiten, ...) und benutzerdefinierten Formatierungen und Anmerkungen zu Lernunterlagen (kurz als Annotationen bezeichnet) möglich (vgl. [Fürlinger, 2003]).

- In der **Library** schließlich werden alle jene digitalen Dateien abgespeichert, welche für eine direkte Speicherung innerhalb von Wissensatomen zu groß sind (z.B. ein Videoclip). In einem solchen Fall enthält das zur digitalen Ressource gehörende Wissensatom lediglich eine Verknüpfung zu einem in der Library gespeicherten BLOB.

In der Folge werden die für die Architektur des Contentpools bedeutenden Komponenten „**Persönlicher Bereich**" und „**Allgemeiner Contentpool**" noch detaillierter beschrieben.

Abbildung 49 (Seite 150) zeigt das Konzept für die Speicherung von persönlichen Inhalten und Annotationen, wo hingegen Abbildung 50 (Seite 151) auf die Details der Gestaltung des allgemeinen Contentpools Bezug nimmt.

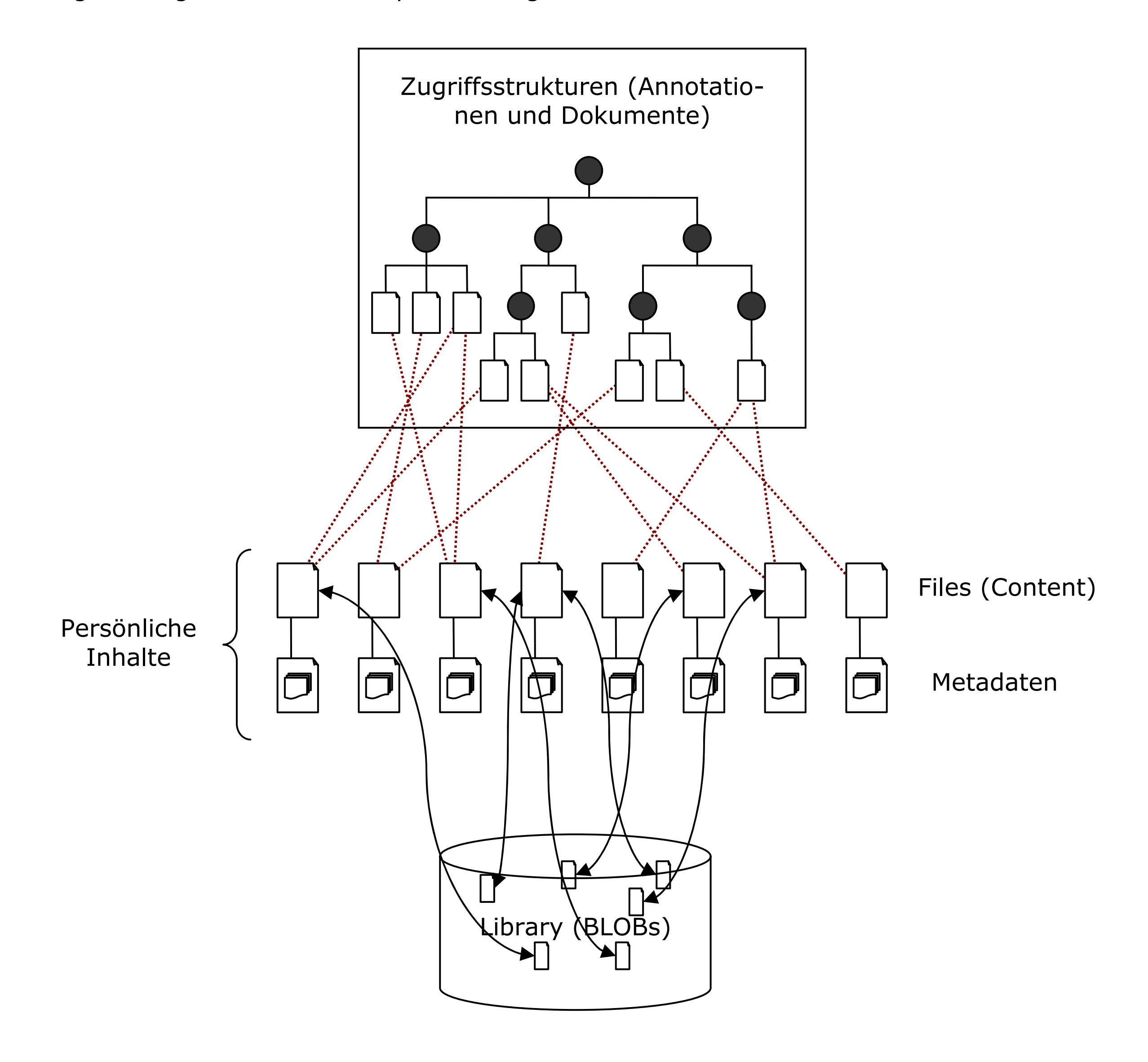


Abbildung 49: Organisation der persönlichen Inhalte – konzeptuelles Modell

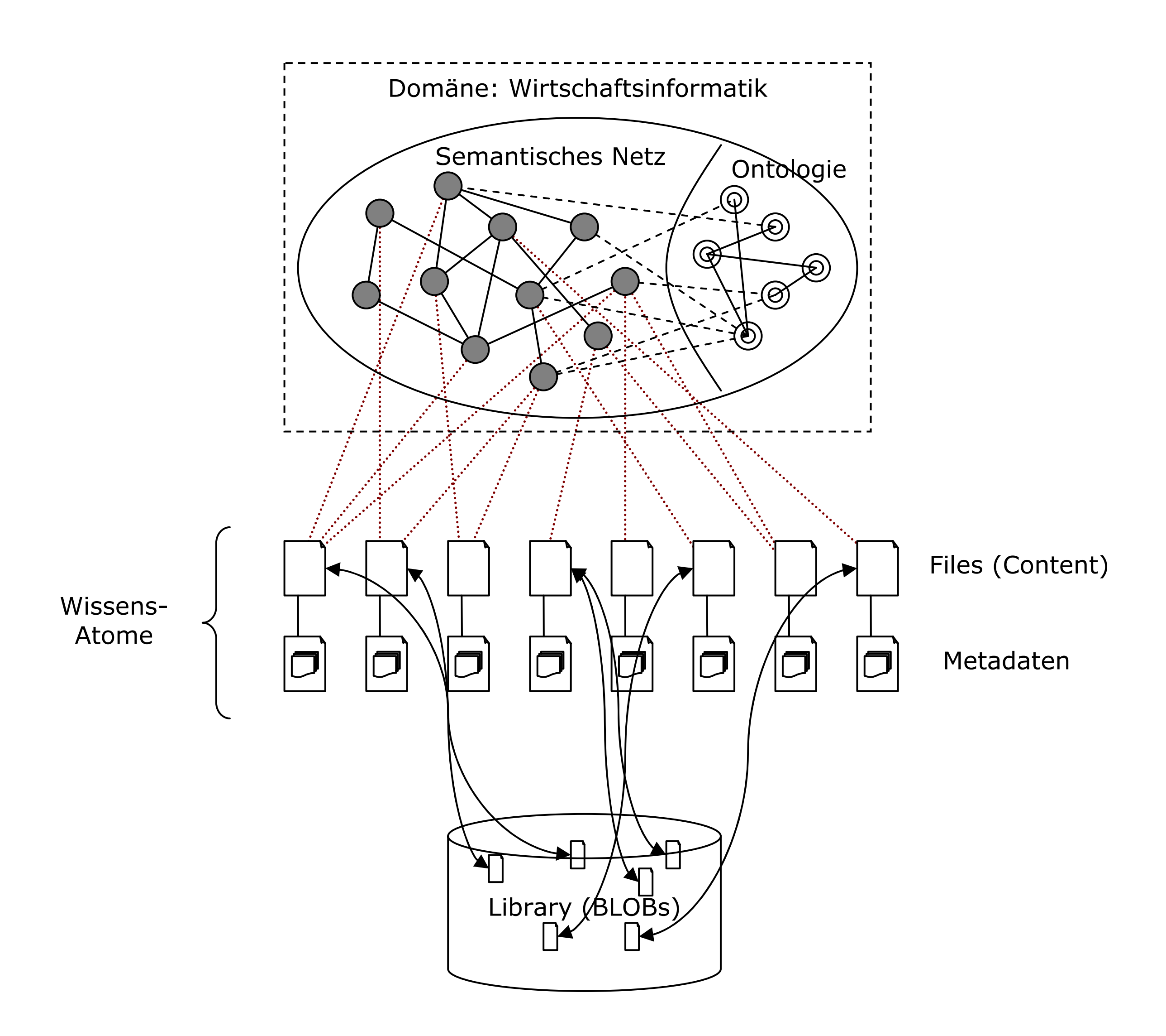


Abbildung 50: Konzeptuelles Modell des allgemeinen Contentpools

Die beiden gezeigten Darstellungen (Abbildung 49 und Abbildung 50) gelten jeweils für ein semantisches Netz im Contentpool. Aufgrund der Tatsache, dass eventuell Bücher aus **verschiedenen Fachgebieten** im Contentpool gespeichert werden sollen, wird auch ein Konstrukt benötigt, mit dessen Hilfe **mehrere semantische Netze** verwaltet werden können.

Dieses Ziel kann mit Hilfe von **Ontologien** erreicht werden. Abbildung 51 zeigt, auf welche Weise semantische Netze mit einer (im System) **globalen, allgemeinen Ontologie** verknüpft werden können. In der allgemeinen Ontologie sind sämtliche Konstrukte gespeichert, welche **in allen Fachgebieten eine Entsprechung** (d.h. einen äquivalenten „Zwillingsbruder") aufweisen. So wird es z.B. in allen wissenschaftlichen Fachgebieten bestimmte Methoden geben, wobei jedoch Methoden der Wirtschaftsinformatik nicht mit Methoden der Psychologie oder der Betriebswirtschaftslehre gleichzusetzen sind und daher in jedem Fachgebiet eigens definiert werden müssen. In der allgemeinen Ontologie entsprechen sich diese Konstrukte hingegen.

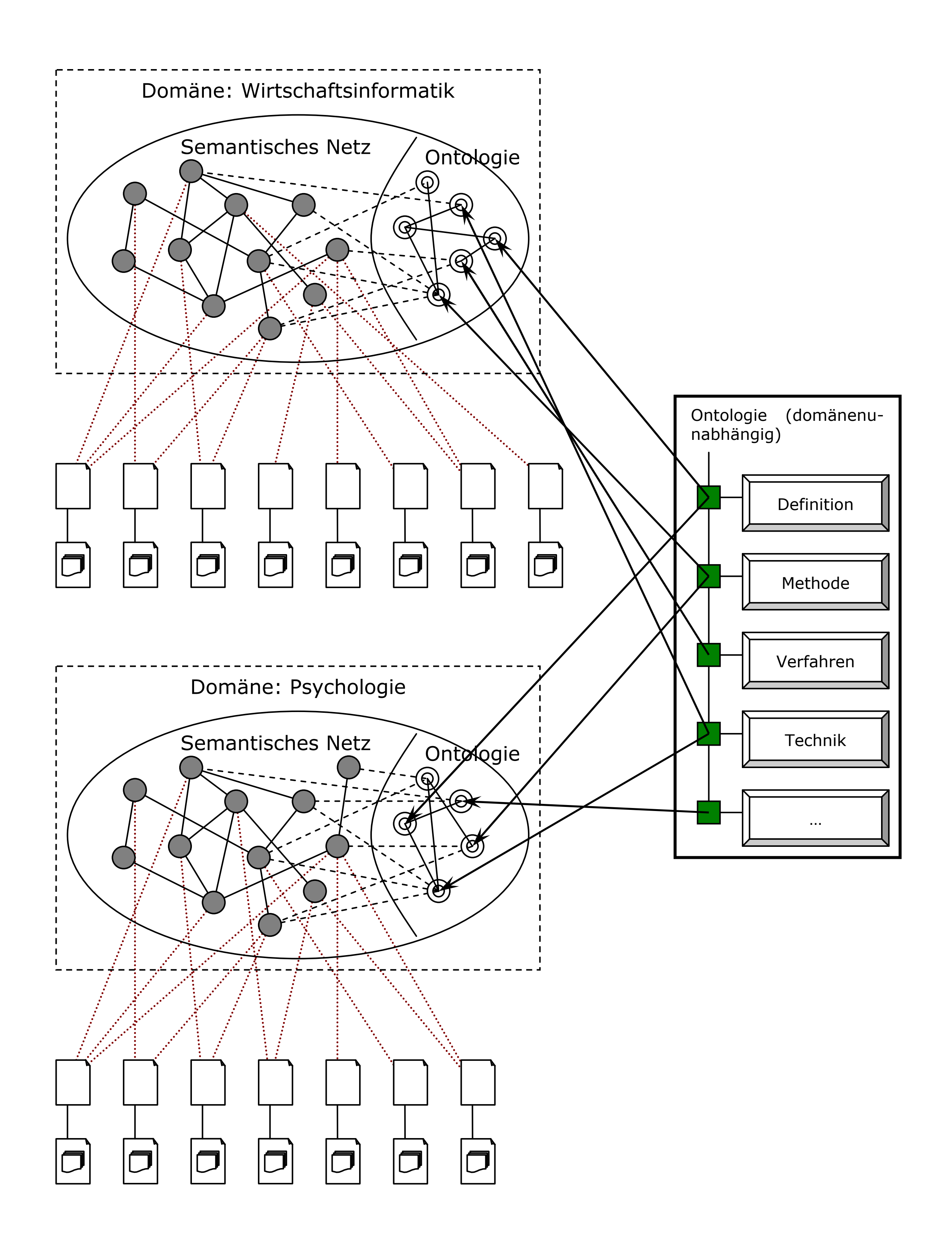


Abbildung 51: Verknüpfung semantischer Netze mit Hilfe von Ontologien

6.1.3 Software- und Verteilungsarchitektur

Während im vorangegangenen Abschnitt 6.1.2 (Konzeptuelle Architektur des Contentpools) das Augenmerk auf der Darstellung der internen Abläufe in der Contentpool-Verwaltung lag, wird in diesem Abschnitt ein Überblick in die Software- und Verteilungsarchitektur gegeben. Zweck ist die Darstellung der **Komponenten** des Systems sowie die Einteilung der Komponenten in **Client- und Serveranwendungen**. Daraus wird zudem die Gestaltung der **Schnittstellen** zwischen den Systemkomponenten ersichtlich.

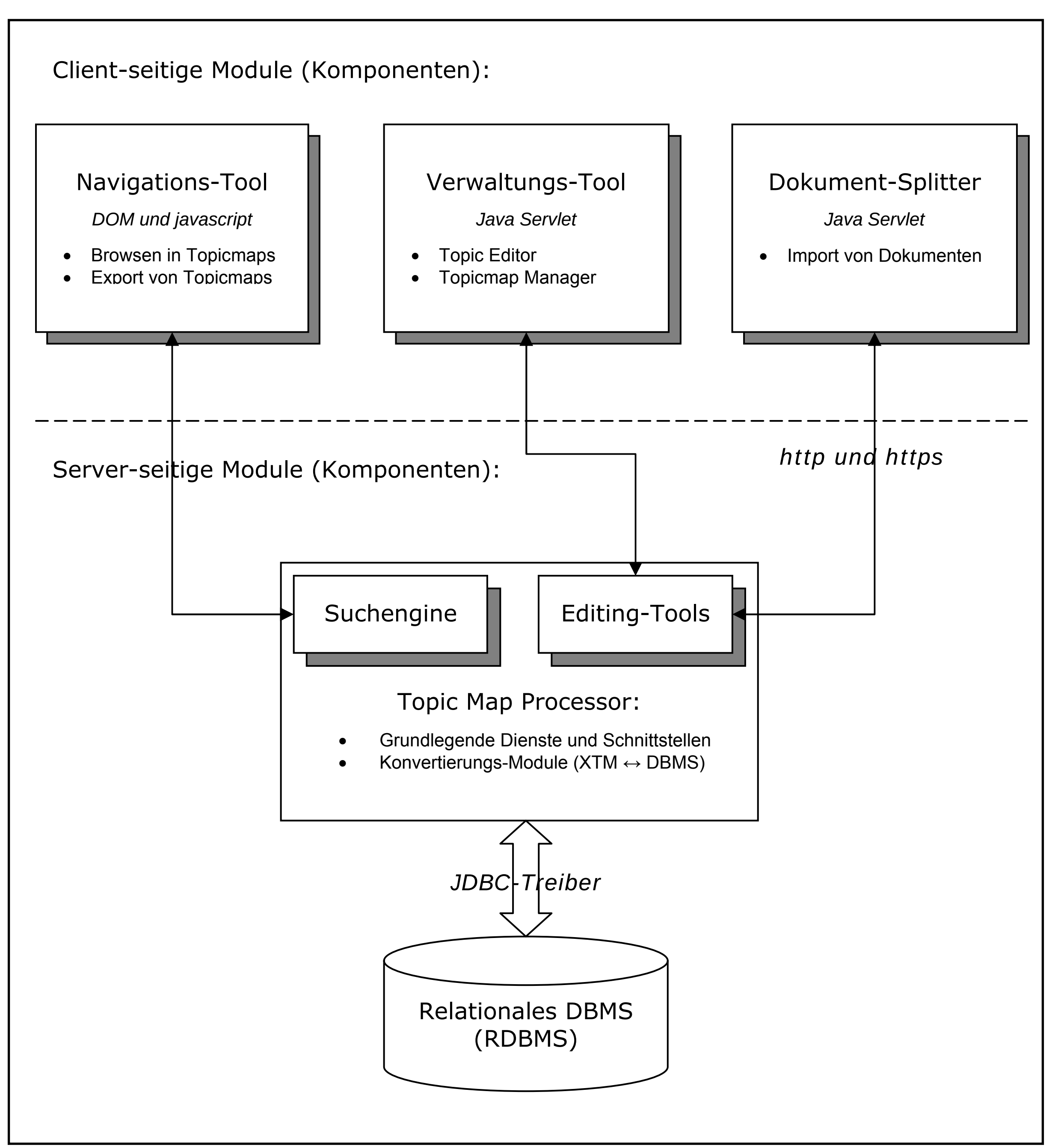


Abbildung 52: Architektur – Scholion WB+ Contentpool-Verwaltung

In Abbildung 52 wird eine Übersicht über die geplante Architektur der Scholion WB+ Contentpool-Verwaltung gegeben. Nachfolgend werden die Teile der Architektur kurz beschrieben, und zwar folgender Maßen unterteilt:

- 6.1.3.1 Server-seitige Komponenten
- 6.1.3.2 Client-seitige Komponenten

Hinweis: Eine wesentlich genauere Definition der Werkzeuge und Spezifikation der genannten Komponenten erfolgt im Abschnitt 6.2 (Modellierung der Prozesse im System) im Rahmen der Vorstellung der Prozessmodelle. Weiters kann im Kapitel 7: Implementierung der Software in den Abschnitten 7.1 (Klassenmodell der Contentpool Verwaltung, Seite 249 ff.) und 7.2 (Klassenmodell des Dokument Splitters, Seite 274 ff.) an Hand der Klassenmodelle die Implementierung des spezifizierten Designs nachvollzogen werden.

6.1.3.1 Server-seitige Komponenten

- Die zentrale Rolle nimmt der so genannte **Topic Map Processor** ein. Dieses Werkzeug dient dazu, die Verbindung zwischen den Client-seitigen Tools (Komponenten) und dem Datenbankmanagementsystem herzustellen. Wichtigste Aufgabe ist die **Konvertierung** der in einem relationalen Datenbankschema gespeicherten Topic Map in ein XML-basiertes Topic Map Format bzw. das Einlesen der **XML Topic Map in Java-Objekte**, mit denen die Module der Contentpool-Verwaltung arbeiten können. Die konvertierte Topic Map wird für die Darstellung mittels DOM und JavaScript zu Grunde gelegt. Der Topic Map Processor stellt zudem Dienste und Schnittstellen für die Suche in der Topic Map sowie für die Bearbeitung der Topic Map und die Bearbeitung der Wissensatome im Contentpool zur Verfügung.
- Im **relationalen Datenbankmanagementsystem** erfolgt die Speicherung aller Daten der Contentpool-Verwaltung. Das DBMS wird auch vom Scholion WB+ Client-Frontend (dem anderen Teil des Projekts, Anm.) verwendet und enthält sämtliche Daten zum semantischen Netz bzw. zu den semantischen Netzen sowie die **eigentlichen Inhalte** in Form von Wissensatomen. Die Kommunikation zwischen DBMS und Topic Map Processor erfolgt mit Hilfe eines JDBC-Treibers.

6.1.3.2 Client-seitige Komponenten

- Das **Navigations-Tool** dient zur benutzerfreundlichen **Erschließung der Inhalte** im semantischen Netz. Der Benutzer hat die Möglichkeit, die Schnittstelle zur Suche zu nutzen, welche vom Topic Map Processor zur Verfügung gestellt wird. Zudem können Teile eines semantischen Netzes exportiert werden, wobei geplant ist, verschiedene Dateiformate zu unterstützen (z.B. HTML, XML, PDF, usw.).
- Mit Hilfe des **Verwaltungs-Tools** können die Inhalte des Contentpools und die semantischen Beziehungen verwaltet werden, so dass gewährleistet ist, den Contentpool auf dem aktuellsten Stand des Wissens halten zu können.
- Aufgabe des **Dokument-Splitters** ist es, Standard-Lehrbücher, welche in einem elektronischen Format vorliegen müssen, in Wissensatome zu zerlegen. Darin ist

sowohl die Erstellung der Wissensatome als auch ihre Einordnung in das semantische Netz des entsprechenden Fachgebiets inbegriffen.

- Die **Kommunikation** zwischen den Client-Komponenten und den Server-Komponenten erfolgt mit Hilfe des http-Protokolls. Bei sensiblen Daten kann auch über SSL, d.h. mit dem https-Protokoll, kommuniziert werden.

6.1.4 Klassenmodell für die Contentpool-Verwaltung

In diesem Abschnitt wird die Darstellung der Architektur der Scholion WB+ Contentpool-Verwaltung weiter präzisiert, indem die Hierarchie aller benötigten Klassen und ihre Beziehungen zueinander entworfen werden. Zudem wird noch die Einordnung der Contentpool-Verwaltung in die Gesamtheit der Wissenstransfer-Anwendung Scholion WB+ gezeigt. Das Klassenmodell wird in folgenden sieben Teilen präsentiert:

6.1.4.1 Gesamtübersicht Scholion WB+

In Abbildung 53 wird die Architektur für das Projekt Scholion WB+ gezeigt. Am linken Rand der Abbildung befinden sich Klassen, die keinem eigenen Package zugeordnet werden sollen; am rechten Rand befinden sich Packages.

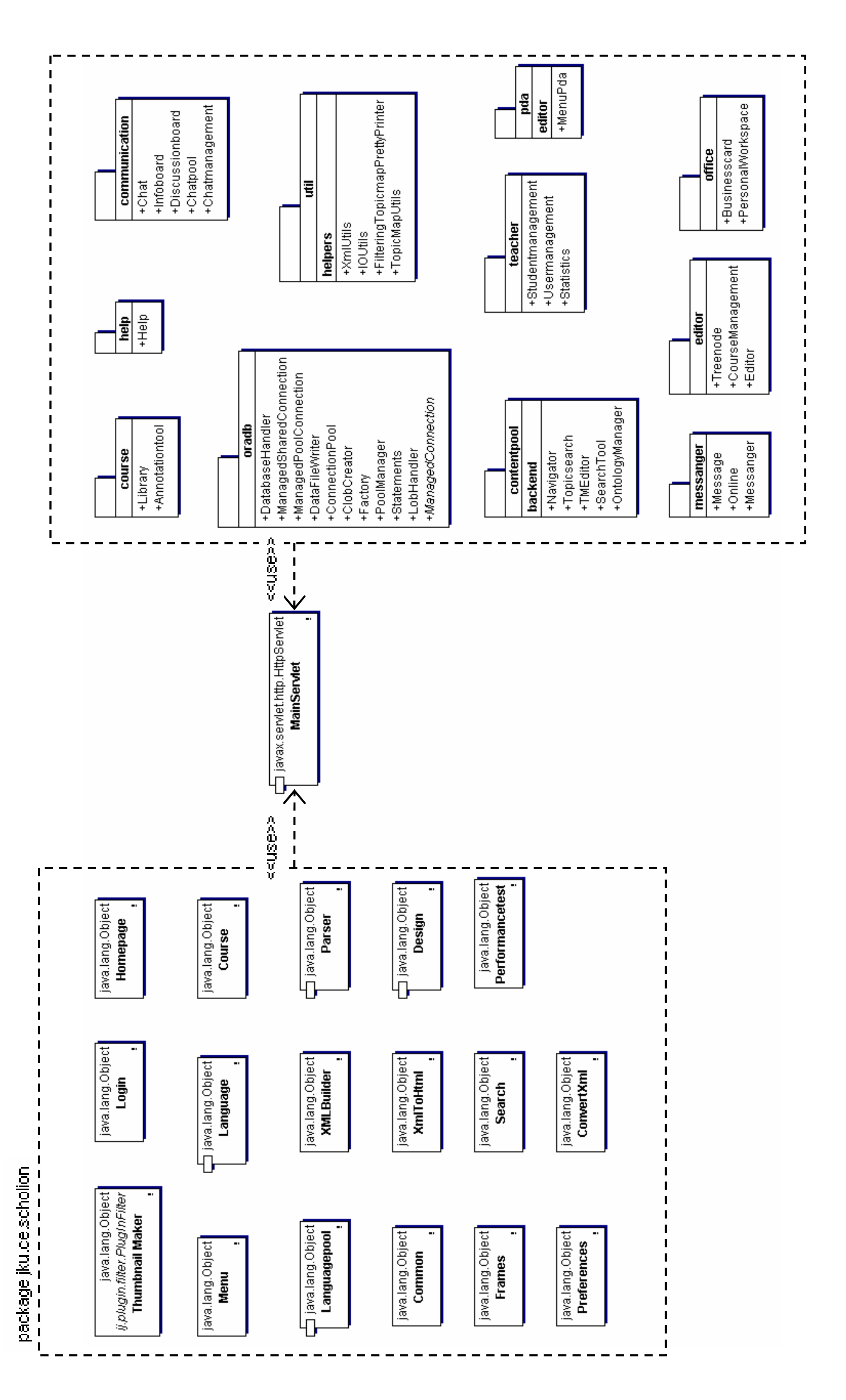


Abbildung 53: Gesamtkonzept für die Systemarchitektur – Scholion WB+

Die Elemente der Architektur in Abbildung 53 können wie folgt kurz erläutert werden:

- Das zentral dargestellte `MainServlet` nimmt die wichtigste Rolle in der Systemarchitektur ein. Das `MainServlet` übernimmt die **Kommunikation mit den Clients** und leitet deren Anfragen an jene Klasse oder jenes Package weiter, die oder das für die jeweils gewünschte Funktionalität die Verarbeitungslogik enthält.
- Der rechte Rand der Abbildung zeigt so genannte **Packages** (welche weitere Klassen logisch zusammen fassen). Jedes Package enthält Klassen, in denen die **Anwendungslogik** für eine bestimmte, genau **definierte Funktionalität** (die einer Anforderung an das System entspricht) implementiert. Für die weiteren Ausführungen in der Diplomarbeit sind innerhalb der Architektur von Scholion WB+ nur die Packages `scholion.contentpool` und `scholion.util` von Interesse.[32] In ihm sind alle Algorithmen zur Verwaltung des Contentpools innerhalb von Scholion WB+ gemäß der Definition der Anforderungen implementiert.
- Am linken Rand der Abbildung werden Klassen dargestellt (wie z.B. `Common`, `Language`, `Login`, `ThumbnailMaker`, …), deren Funktionalität **für alle Packages** im rechten Teil der Abbildung **von Bedeutung** ist. Aus diesem Grund sind diese Klassen selbst keinem Package zugeordnet, sondern befinden sich direkt im Package `scholion`.

Aus der obigen Abbildung wird zudem ersichtlich, wie die Contentpool-Verwaltung in die Architektur des Projekts Scholion WB+ eingegliedert wird. Wie weiter oben erwähnt, nimmt das `MainServlet` die bedeutendste Rolle in Scholion WB+ ein. Funktionalität, die sich logisch zusammen fassen lässt, wird jeweils einem eigenen Package zugeordnet und in mehreren (meist ungefähr 5 – 10) Klassen implementiert.

In diesem Sinn wurde für die **Contentpool-Verwaltung** innerhalb der Architektur von Scholion WB+ das **Package `scholion.contentpool`** vorgesehen. Das Package `scholion.contentpool` bindet Algorithmen zur Verwaltung von Topic Maps und zur Verwaltung von Wissensatomen in die Wissenstransfer-Umgebung Scholion WB+ ein.

Teile der Funktionalität der Contentpool-Verwaltung befinden sich darüber hinaus im Package `scholion.util`. Da die Funktionalität der Klassen im zuletzt genannten Package auch von anderen Teilen von Scholion WB+ genutzt wird, wurden die gemeinsam genutzten Klassen im Package `scholion.util` eingeplant.

6.1.4.2 Übersicht: Klassen und Packages der Contentpool-Verwaltung

Die Contentpool-Verwaltung wird in Übereinstimmung mit dem in Abbildung 52 (siehe Seite 153) gezeigten Design in mehreren Klassen implementiert, welche in drei Packages zusammen gefasst sind: `scholion.contentpool`, `scholion.util` und `utils.topicmaps`.

---

[32] **Hinweis**: grundlegende Funktionalität für die Verwaltung von Topicmaps, die für die Contentpool-Verwaltung unverzichtbar sind, aber mit Scholion WB+ in keinem unmittelbaren Zusammenhang stehen, sind im Package `utils.topicmaps` implementiert. Vgl. dazu den Abschnitt 6.1.4.2 (Übersicht: Klassen und Packages der Contentpool-Verwaltung, Seite 157), Abbildung 56: Klassen und Werkzeugklassen für die Verwaltung von Topic Maps.

Auf den folgenden Seiten zeigt die Abbildung 54 das Package `scholion.contentpool`, die Abbildung 55 das Package `scholion.util`, die Abbildung 56 das Package `utils.topicmaps`.

Die genannten Packages erfüllen folgende Aufgaben:

- Das Package `scholion.contentpool` ist für die **Integration grundlegender Algorithmen** der Verwaltung von Topic Maps sowie der Verwaltung von Wissensatomen in die Wissenstransfer-Umgebung Scholion WB+ verantwortlich. In ihm ist der Hauptteil der Funktionalität „Contentpool-Verwaltung" im Kontext von Scholion WB+ implementiert.
- Das Package `scholion.util` fasst die von allen Scholion-Packages benötigte Funktionalität in Klassen und **Werkzeug-Klassen** zusammen. Auf diese Weise werden wichtige Algorithmen der **gesamten Wissenstransfer-Umgebung** Scholion WB+ zur Verfügung gestellt.
- Im Package `utils.topicmaps` schließlich sind die grundlegenden Algorithmen zur **Verwaltung von Topic Maps** implementiert. Diese Klassen sind jedoch nicht primär für den Einsatz mit Scholion WB+ entworfen, sie können vielmehr in jeder beliebigen Anwendung als **Klassen-Bibliothek** zum Einsatz kommen. Mit anderen Worten, die Klassen im Package `utils.topicmaps` stellen eine **Schnittstelle** zur Manipulation von Topic Maps auf sehr **hohem Abstraktionsniveau** zur Verfügung, sind dabei aber unabhängig von einem bestimmten Kontext implementiert.

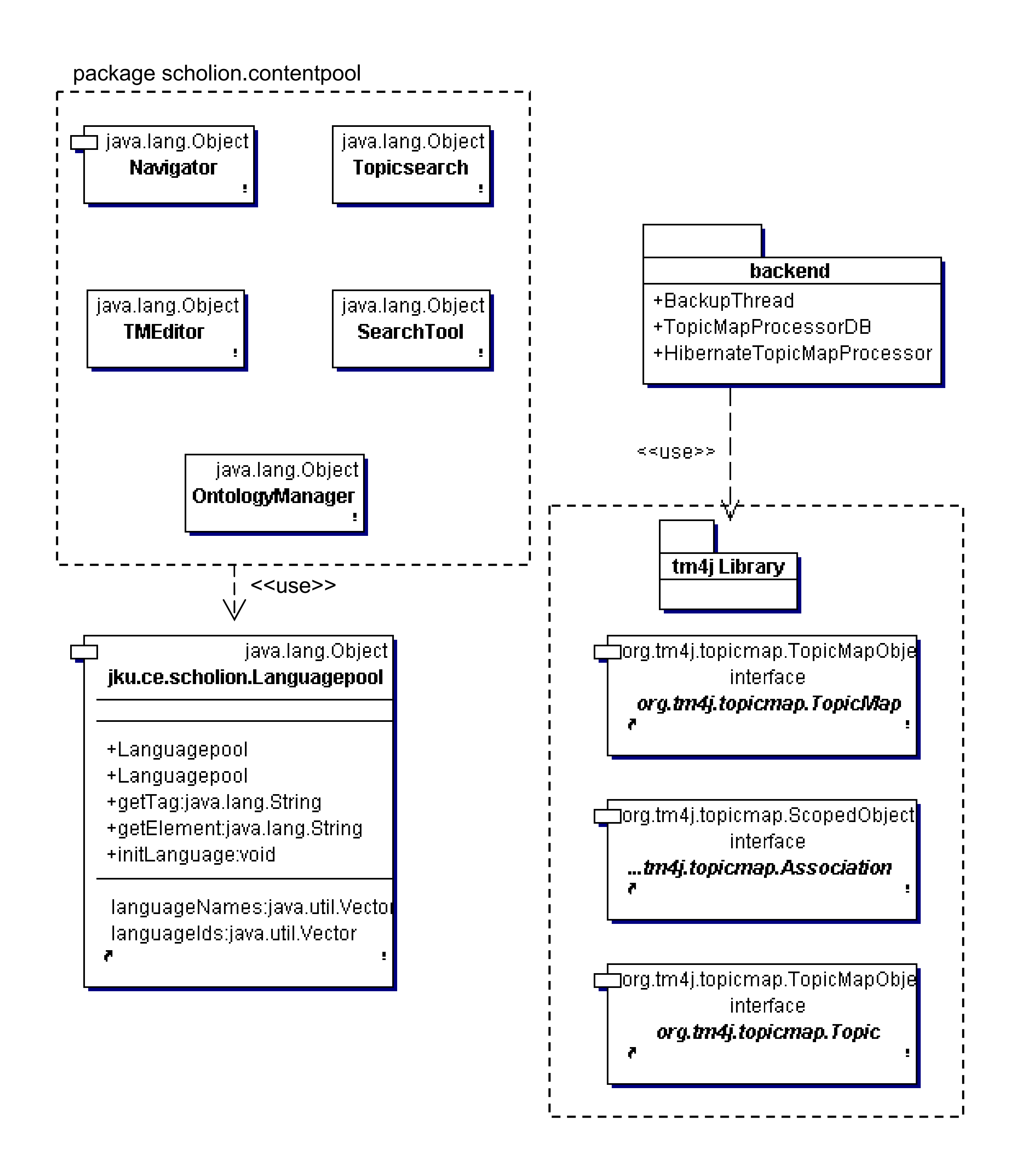


Abbildung 54: Contentpool-Verwaltung: Klassen und Packages

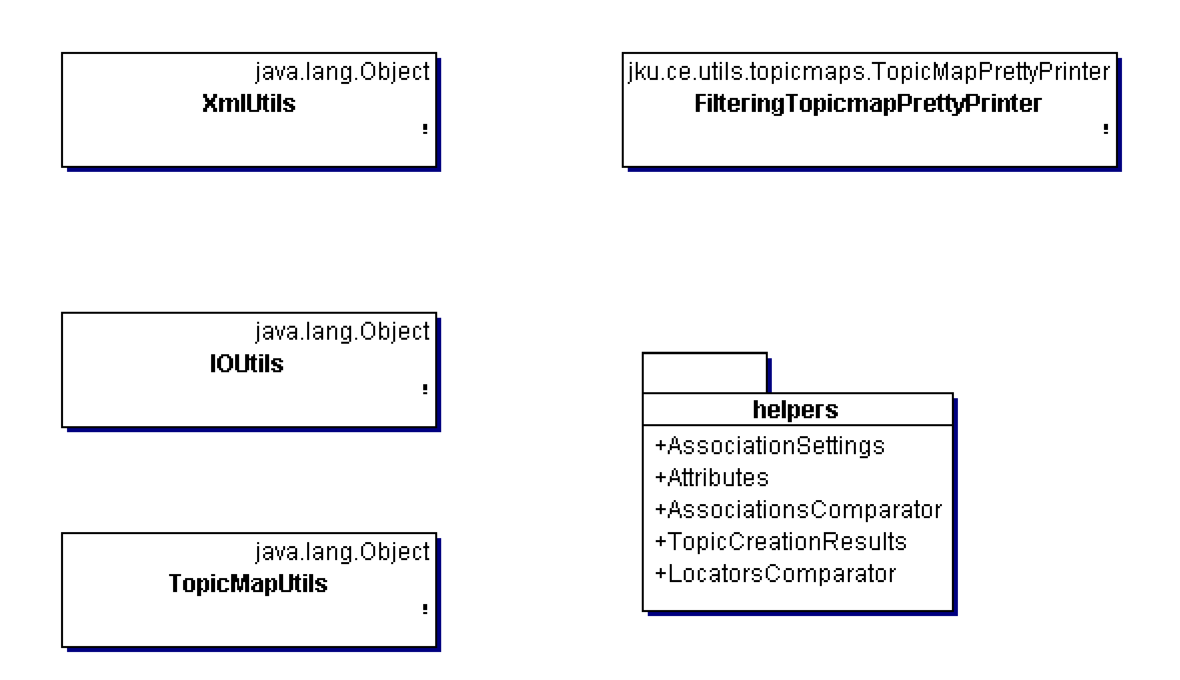


Abbildung 55: Contentpool-Verwaltung: Werkzeug-Klassen im Package „util"

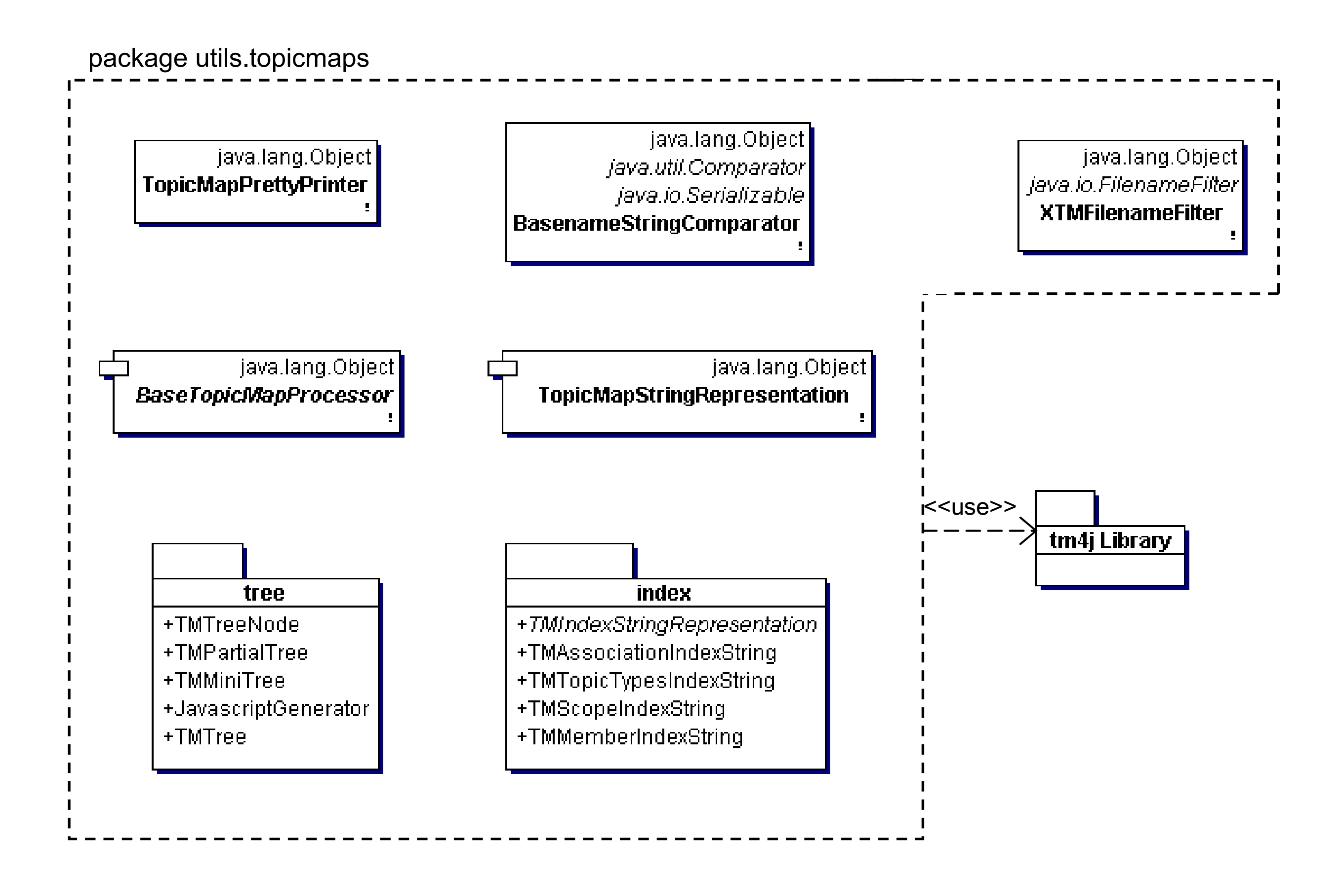


Abbildung 56: Klassen und Werkzeugklassen für die Verwaltung von Topic Maps

Die oben gezeigten Abbildungen geben einen groben Überblick der Architektur für die Contentpool-Verwaltung. Auf den folgenden Seiten geht der Autor auf die einzelnen Teile der Contentpool-Verwaltung (Navigation, Verwaltung und Erstellung von Wissensatomen) bis auf die Detailebene einzelner Klassen ein. Zunächst erfolgt noch die genaue Spezifikation des in Abbildung 56 gezeigten Packages für die Verwaltung von Topic Maps.

6.1.4.3 Schnittstelle (Algorithmen) für die Verwaltung von Topic Maps

Wie im vorangegangenen Abschnitt 6.1.4.2 (Übersicht: Klassen und Packages der Contentpool-Verwaltung) erwähnt, sind im Package `utils.topicmaps` die grundlegenden Algorithmen für die Verwaltung von Topic Maps implementiert. Die Klassen des Packages stellen somit eine **Schnittstelle** zur Verfügung, mit der **Topic Maps** in komfortabler Weise **bearbeitet** werden können.

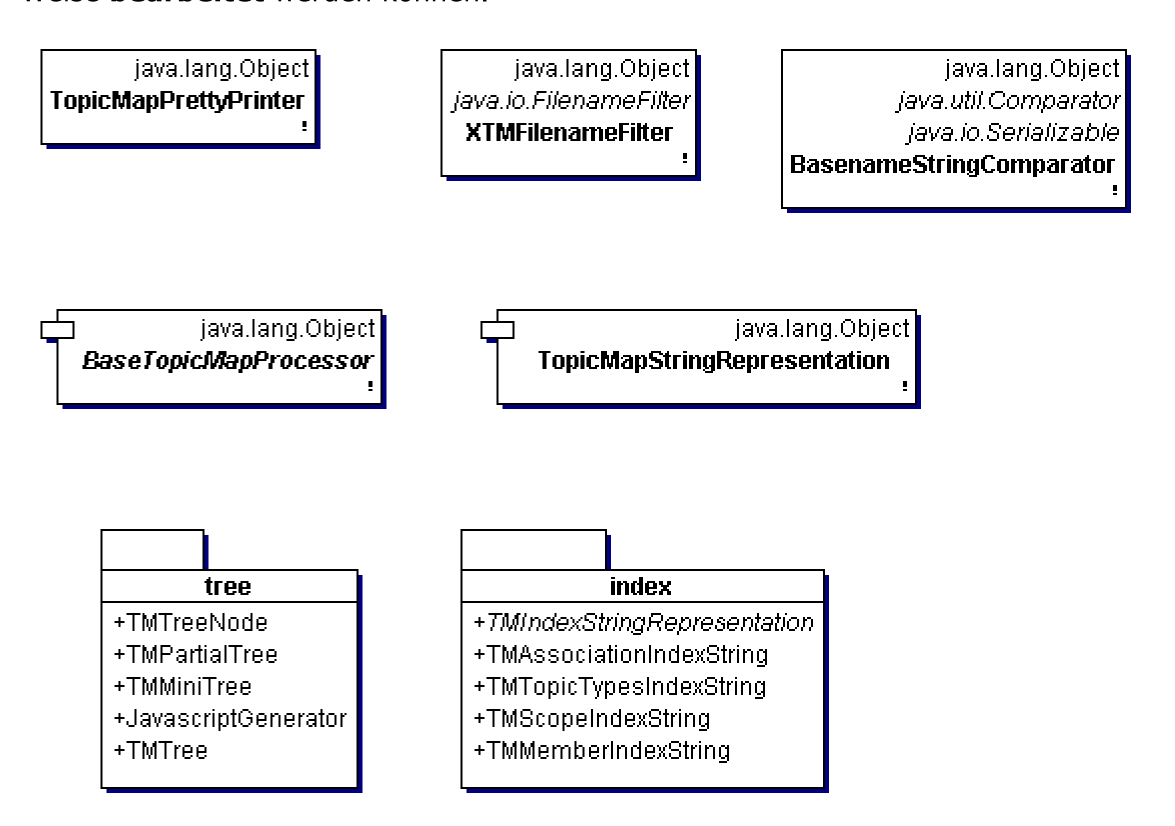


Abbildung 57: Klassen zur Bearbeitung von Topic Maps im Package utils.topicmaps

Die Funktionalität der in Abbildung 57 gezeigten Klassen und Sub-Packages lässt sich wie folgt kurz beschreiben:

- `BaseTopicMapProcessor` ist die wichtigste Klasse im Package. Sie stellt **Dienste** zur Verfügung, mit denen auf Topic Maps oder die Objekte in Topic Maps zugegriffen werden kann. Die Klasse `BaseTopicMapProcessor` arbeitet eng mit der tm4j Library [TM4J, 2002] zusammen.
- `TopicMapPrettyPrinter` dient dazu, eine für die Ansicht auf dem Bildschirm oder für den Ausdruck auf Papier geeignete **textuelle Darstellung** einzelner Topics aus einer Topic Map zu erzeugen.

- `TopicMapStringRepresentation` **berechnet** aus einer Topic Map die wichtigsten **Metadaten**. Dazu zählen eine Liste der enthaltenen Topics, statistische Angaben (Anzahl der Associations, Anzahl der Occurrences, Anzahl der Topic Types, usw.).
- `BasenameStringComparator` implementiert einen Algorithmus, mit dessen Hilfe zwei `Basename`s auf **Gleichwertigkeit überprüft** werden können.
- `XTMFilenameFilter` ist eine Hilfsklasse, mit der **Topic Map Dateien** im Dateisystem an Hand ihrer Datei-Endung erkannt werden können.
- Im Sub-Package `tree` sind Klassen enthalten, die Algorithmen für eine auf der Script-Sprache **JavaScript** basierende, in Form eines Baumes strukturierte Darstellung einer Topic Map implementieren.
- Im Sub-Package `index` werden Klassen zusammen gefasst, die eine **textuelle Darstellung eines Index** auf eine Topic Map erzeugen können. Auf diese Weise kann Information zu einer Topic Map (z.B. Anzahl, Namen und semantische Bedeutung von Scopes) in textueller Form dargestellt werden.

6.1.4.4 Topicmap-Processor (Package contentpool.backend)

Der so genannte Topicmap-Processor (vgl. Abbildung 52: Architektur – Scholion WB+ Contentpool-Verwaltung, Seite 153) wird im Package `contentpool.backend` implementiert. Der Topicmap-Processor ist **zu komplex**, als dass er wirklich in einer einzigen Klasse realisiert werden könnte. Aus diesem Grund wurde die **Zerlegung in mehrere Klassen** vorgenommen.

Eine Übersicht der Klassen des Topicmap-Processors sowie der angebotenen Dienste (die der Schnittstelle entsprechen) bietet die Abbildung 54 weiter unten.

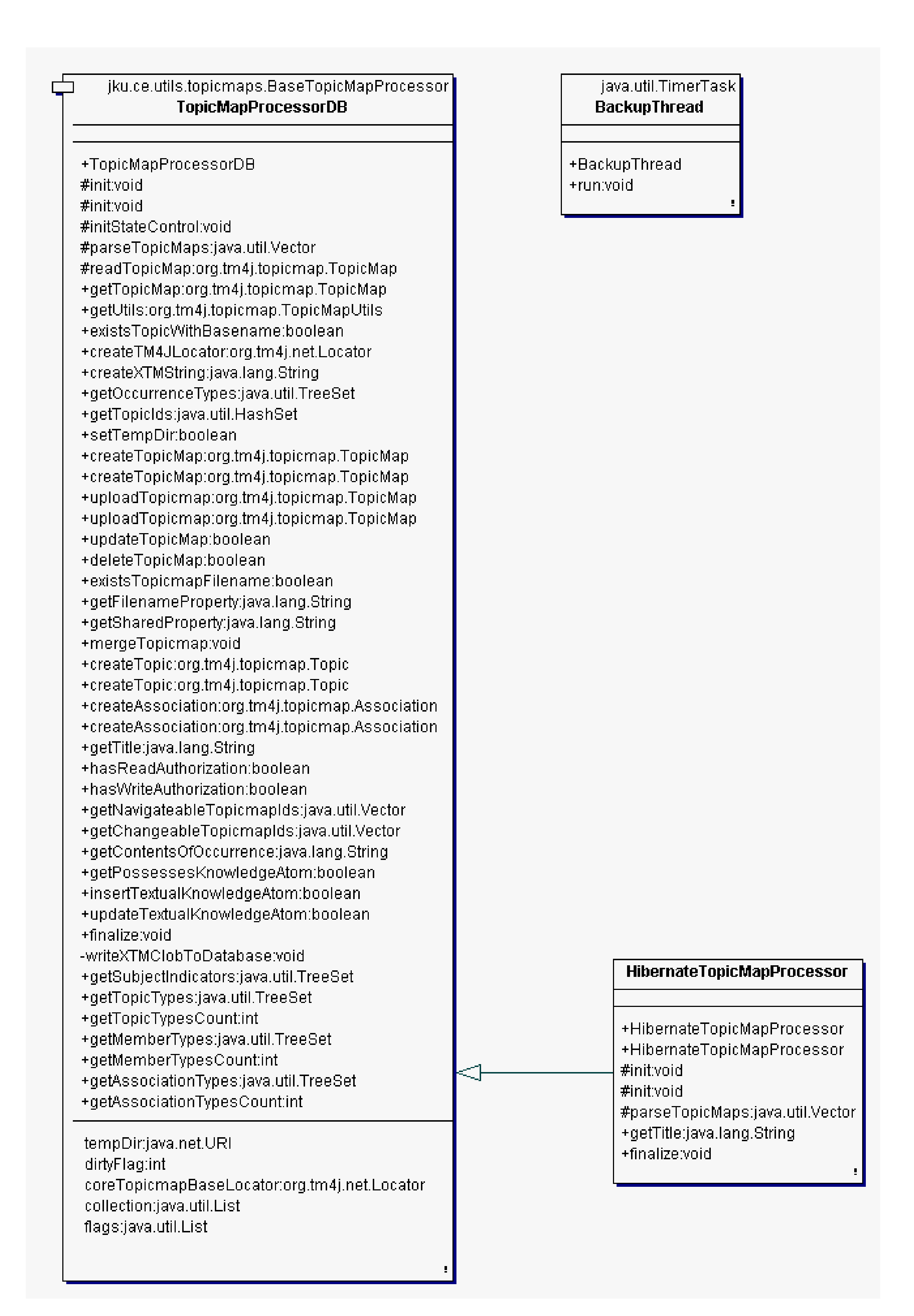


Abbildung 58: Klassenmodell des Topicmap-Processors

Die wichtigsten, von der Klasse `TopicmapProcessorDB` angebotenen Dienste sind die folgenden:

- Manipulation von Topic Maps (Hinzufügen, Bearbeiten und Löschen);
- Manipulation von Objekten in Topic Maps (Hinzufügen, Bearbeiten und Löschen von *Topics*, *Associations*, *Occurrences* und anderen Topicmap Objekten);
- Verschmelzen („Mergen") von Topic Maps;
- Zugriff auf Topic Maps;
- Exportieren von Topic Maps als XML-Dateien, die dem XML Topic Map Schema [TM Strict, 2000] entsprechen.

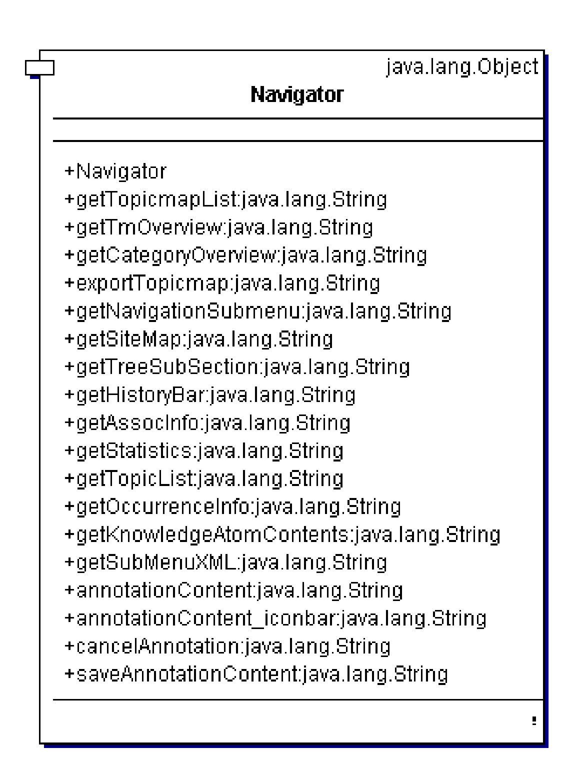
`HibernateTopicmapProcessor` ist eine Klasse, die das von Topic Maps for Java (tm4j) angebotene, so genannte **Hibernate Datenmodell** auf Basis einer **relationalen Datenbank** unterstützt. Kernstück des erwähnten Hibernate Datenmodells ist die automatische Erstellung des relationalen Datenmodells mit Hilfe eines Werkzeugs, das Java-Objekte auf ein Datenbank-Schema abbildet (vgl. [Hibernate, 2003] und [TM4J, 2002]).

`BackupThread` schließlich ist eine Hilfsklasse, die für das regelmäßige **Sichern von Daten** im Topicmap-Processor zuständig ist.

6.1.4.5 Navigations-Tool (Topicmap-Navigator)

Das Navigations-Tool wird benötigt, um in den **Wissensatomen des Contentpools** navigieren zu können. Wie in Abbildung 52 (Architektur – Scholion WB+ Contentpool-Verwaltung, siehe Seite 153) gezeigt, stellt das Navigations-Tool eines von drei **client-seitigen Modulen** in der Architektur der Contentpool-Verwaltung dar.

In Abbildung 59 wird die Architektur des Navigations-Tools als Klassenmodell gezeigt.



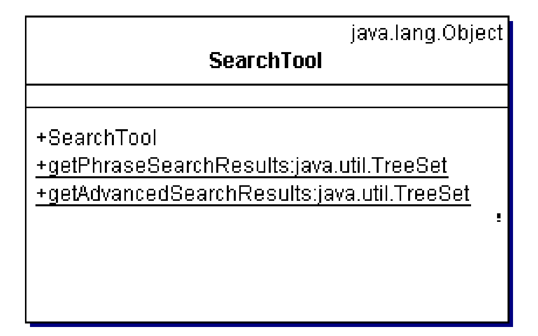


Abbildung 59: Klassenmodell des Navigations-Tools

Wie aus seiner Bezeichnung und Beschreibung hervorgeht, kann mit dem Navigationstool **keine Manipulation** von Wissensatomen oder Topic Maps vorgenommen werden. Die einzelnen Elemente des in Abbildung 59 gezeigten Klassenmodells lassen sich wie folgt kurz beschreiben:

- In der Klasse `Navigator` ist der **Hauptteil der Funktionalität** des Navigations-Tools implementiert. Wie aus Abbildung 59 ersichtlich wird, umfasst die Funktionalität z.B. das Anzeigen von statistischer Information, die Navigation von Topics, Associations und Occurrences, und vieles mehr.
- Die Klassen `Topicsearch` und `SearchTool` ermöglichen eine **Such-Funktion** innerhalb von Topic Maps, wobei in `SearchTool` die Anwendungslogik, in `Topicsearch` hingegen das Generieren der Benutzungsschnittstelle implementiert ist.

6.1.4.6 Verwaltungs-Tool (Topic-Editor, Topicmap-Manager)

Mit Hilfe des Verwaltungs-Tools können **Wissensatome** und **Topic Maps** im Contentpool „verwaltet" werden. Diese Aufgabe umfasst das **Erstellen, Bearbeiten und Löschen** von Wissenatomen, Topic Maps und Objekten in Topic Maps. Wie in Abbildung 52 (Architektur – Scholion WB+ Contentpool-Verwaltung, siehe Seite 153) gezeigt, stellt das Verwaltungs-Tool eines von drei **client-seitigen Modulen** in der Architektur der Contentpool-Verwaltung dar.

In Abbildung 60 ist die Architektur des Verwaltungs-Tools als Klassenmodell dargestellt.

```
java.lang.Object
TMEditor
-----------------------------------------------
+TMEditor
+getNotAuthorizedPage:java.lang.String
+getStartPage:java.lang.String
+getMainMenu:java.lang.String
+getVersionHistory:java.lang.String
+getNewTopicScreen:java.lang.String
+saveNewTopic:java.lang.String
+deleteTopics:java.lang.String
+saveEditTopicProperties:java.lang.String
+saveEditTopicAssociations:java.lang.String
+getNewTopicConfirmScreen:java.lang.String
+getNewTopicFailedScreen:java.lang.String
+getEditTopicScreen:java.lang.String
+getEditTopicPropertiesScreen:java.lang.String
+getEditTopicAssociationsScreen:java.lang.String
+getEditOccurrencesScreen:java.lang.String
+saveEditOccurrences:java.lang.String
+getEditTopicSuccessScreen:java.lang.String
+getEditTopicFailedScreen:java.lang.String
+getDeleteTopicScreen:java.lang.String
+getDeleteTopicConfirmScreen:java.lang.String
+getDeleteTopicSuccessScreen:java.lang.String
+getDeleteTopicFailedScreen:java.lang.String
+getNavigation:java.lang.String
+getSubMenuXML:java.lang.String
+getChooseTopicmapScreen:java.lang.String
+saveTopicmapSetting:java.lang.String
+saveOccContents:java.lang.String
+getTopicClassification:java.lang.String
+getOccurrenceContentsXML:java.lang.String
```

```
java.lang.Object
OntologyManager
-----------------------------------------------
+OntologyManager
+getNotAuthorizedPage:java.lang.String
+getStartPage:java.lang.String
+getMainMenu:java.lang.String
+getNewTopicmapScreen:java.lang.String
+saveNewTopicmap:java.lang.String
+saveEditTopicmap:java.lang.String
+saveDeleteTopicmap:java.lang.String
+getNewTopicmapConfirmScreen:java.lang.String
+getEditTopicmapScreen:java.lang.String
+getEditTopicmapConfirmScreen:java.lang.String
+getDeleteTopicmapScreen:java.lang.String
+getDeleteTopicmapConfirmScreen:java.lang.String
+getVersionHistory:java.lang.String
+getUploadScreen:java.lang.String
+getUploadFailureScreen:java.lang.String
+saveUpload:java.lang.String
+getDeallocationScreen:java.lang.String
+saveDeallocation:java.lang.String
+getDeallocationFailureScreen:java.lang.String
+getMergingScreen:java.lang.String
+saveMerging:java.lang.String
+getMergingFailureScreen:java.lang.String
+getChooseTopicmapScreen:java.lang.String
+saveTopicmapSetting:java.lang.String
```

Abbildung 60: Klassenmodells des Verwaltungs-Tools (Topic-Editor, Topicmap-Manager)

Die Elemente des in Abbildung 60 gezeigten Klassenmodells lassen sich wie folgt kurz beschreiben:

- Die Klasse `OntologyManager` ist für die **Verwaltung von Topic Maps** verantwortlich. Die Bezeichnung wurde deshalb so gewählt, weil in der Verwaltung der Topic Maps die Verwaltung der Ontologien inkludiert ist. (Zum Begriff der „Ontologie" im Kontext der Contentpool-Verwaltung vgl. den Abschnitt 6.1.2 Konzeptuelle Architektur des Contentpools, Seite 147 ff., sowie die Beschreibung zur Abbildung 92: Modell des Topicmap-Managers, Seite 206 im Abschnitt 6.2 Modellierung der Prozesse im System.) Im `OntologyManager` ist die **Anwendunglogik** zum Erstellen, Bearbeiten und Löschen von Topic Maps als Gesamtheit implementiert.
- Die Klasse `TMEditor` (Topic Map Editor) ist für die **Verwaltung von Objekten** in Topic Maps verantwortlich. Im `TMEditor` ist die Anwendunglogik zum Erstellen, Bearbeiten und Löschen von Objekten in Topic Maps (z.B. Topics, Associations und Occurrences, vgl. die Ausführungen im Abschnitt 4.4.2 XML Topic Maps (XTM), Seite 77 ff.) sowie zum Erstellen, Bearbeiten und Löschen von Wissensatomen (als Occurrences der Topics in Topic Maps) implementiert.

6.1.4.7 Dokument-Splitter

Der Dokument-Splitter dient zum **Zerlegen von Dokumenten** in elektronischer Form und damit zum Erstellen von Objekten (Wissensatomen) im Contentpool.

Die Architektur des Dokument-Splitters als Klassenmodell wird in Abbildung 61 dargestellt:

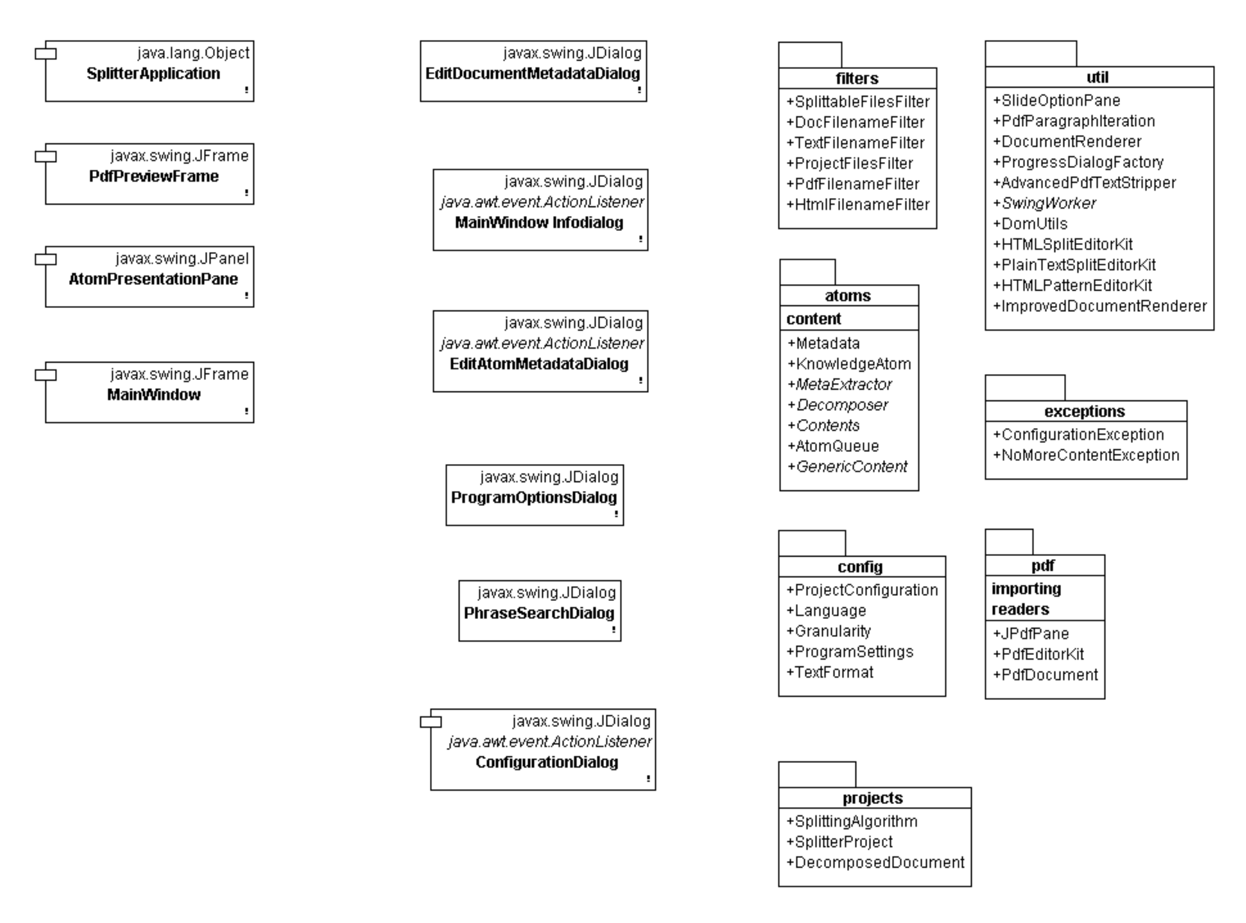


Abbildung 61: Klassenmodell des Dokument-Splitters

Wie in Abbildung 52 (Architektur – Scholion WB+ Contentpool-Verwaltung, siehe Seite 153) gezeigt, stellt der Dokument-Splitter eines von drei **client-seitigen Modulen** in der Architektur der Contentpool-Verwaltung dar. Der Dokument-Splitter wird als eigene **Applikation mit grafischer Benutzungsschnittstelle** (GUI) implementiert, wobei die Ergebnisse der Zerlegung eines Dokuments im Contentpool integriert werden können.

Die Elemente des in Abbildung 61 gezeigten Klassenmodells können wie folgt kurz beschrieben werden:

- `SplitterApplication` stellt die wichtigste Klasse dar. In dieser Klasse ist die Anwendungslogik zum **Start der Applikation** und zum **Ausführen** des Dokument-Splitters und zur **Steuerung** des Dokument-Splitters während seiner Ausführung implementiert.
- Algorithmen und Anwendungslogik zum Import und zum Zerlegen von Dokumenten in elektronischer Form sind in **mehreren Packages und Sub-Packages**

implementiert, die von `SplitterApplication` verwendet werden. Folgende Packages werden für den Dokument-Splitter benötigt:

- Package `projects`, das Klassen für die Verwaltung von **Projekten** im Dokument-Splitter enthält.
- Package `config`, in dem sich Klassen zur Speicherung und Verwaltung der **Konfiguration** des Dokument-Splitters befinden.
- Package `filters`, das Klassen und Anwendungslogik für die **Filterung von Dateien** enthält. Diese Filterung dient dazu, elektronische Dokumente zu identifizieren, die für die Zerlegung in Wissensatome in Frage kommen.
- Package `pdf`, in dem Klassen und Anwendungslogik für das Parsen, den Import und die Darstellung von elektronischen **Dokumenten im PDF-Format** gruppiert sind. Besonders die Aufgabe der Darstellung von Dokumenten im PDF-Format ist alles andere als trivial. Vgl. dazu die Ausführungen im Kapitel 9 Fazit und Ausblick, Seite 314 ff.
- Package `atoms`, das Klassen für die Speicherung von **Wissensatomen**, die durch Zerlegung eines elektronischen Dokuments gewonnen wurden, enthält.
- Package `util`, in dem für die Funktion des Dokument-Splitters wichtige **Utility-Klassen** gesammelt sind.
- Package `exceptions`, das verschiedene Exceptions zur **Beschreibung von Fehlersituationen**, die während der Ausführung des Dokument-Splitters auftreten können, enthält.

- Die grafische Benutzungsschnittstelle schließlich ist den Klassen `MainWindow`, `AtomPresentationPane`, `PdfPreviewFrame`, und den zahlreichen Subklassen von `javax.swing.JDialog` implementiert. In diesen Klassen ist keine Anwendungslogik des Dokument-Splitters realisiert.

6.2 Modellierung der Prozesse im System

In diesem Kapitel erfolgt die genaue Spezifikation der Tools der Scholion WB+ Contentpool-Verwaltung in Form von Modellen der im System ablaufenden Prozesse. Es werden alle Komponenten, welche im Kapitel 6.1 (Systemarchitektur: Scholion WB+) vorgestellt wurden, auf diese Weise präziser spezifiziert.

Zuerst erfolgt eine Vorstellung der verwendeten **Notation** (Abschnitt 6.2.1 Notation für die Prozessmodellierung). Anschließend werden die Prozesse im System vorgestellt, und zwar unterteilt in Prozesse zur **Interaktion zwischen System und Benutzer** (Abschnitt 6.2.2 Interaktionsprozesse) sowie systeminterne Prozesse (Abschnitt 6.2.3 Systemprozesse). Abschließend wird eine Übersicht gegeben, welche **Berechtigungen** ein Benutzer benötigt, um die Contentpool-Verwaltung starten und ihre Werkzeuge verwenden zu dürfen (Abschnitt 6.2.4 Berechtigungsbaum für die Contentpool-Verwaltung).

6.2.1 Notation für die Prozessmodellierung

Die Symbole zur Notation der Prozessmodelle für den Scholion WB+ Contentpool orientieren sich an der in [Auinger, 1999] (Seite 152 ff.) verwendeten Notation. In der folgenden Abbildung 62 sind die verwendeten Symbole erschöpfend angeführt.

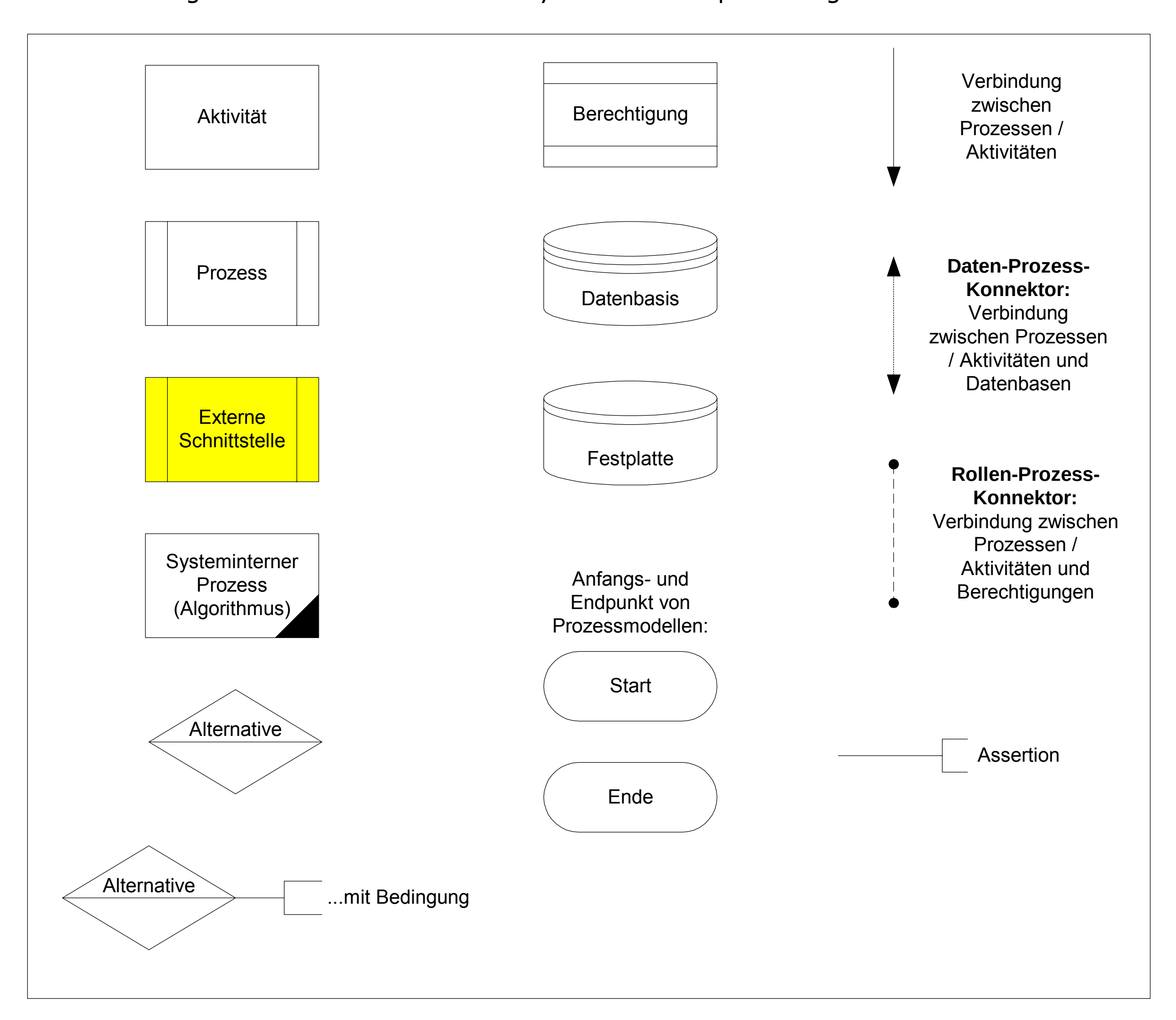


Abbildung 62: Notation (Symbole) für die Prozessmodellierung (nach [Auinger, 1999])

Nachfolgend werden die in Abbildung 62 gezeigten Symbole kurz erläutert.

- **Aktivitäten** sind Systemaktivitäten, die durch Interaktion des Benutzers mit dem System ausgelöst wurden. Dabei wird (in den Modellen) stets angenommen, dass die Aktivität vom Benutzer **bewusst und absichtlich** ausgelöst wurde.
- Ein **Prozess** ist eine Zusammenfassung (Abstraktion) mehrerer Aktivitäten. D.h. für jeden Prozess gibt es ein weiteres Modell, in dem eine **Verfeinerung** bis auf Ebene der Aktivitäten erfolgt.

- Eine **externe Schnittstelle** liegt dann vor, wenn durch eine Benutzerinteraktion ein Prozess ausgelöst wird (z.B. ein Tool wird gestartet), der außerhalb der Tools und Features der Contentpool-Verwaltung liegt. Dies hat zur Folge, dass der damit modellierte Prozess nicht innerhalb des vorliegenden Dokuments verfeinert wird, sondern innerhalb der Prozessmodelle des Scholion WB+ Client-Frontends. Wie bei den Aktivitäten (siehe weiter oben) wird in den Modellen stets angenommen, dass der externe Prozess vom Benutzer **bewusst und absichtlich** ausgelöst wurde.
- **Interne Systemprozesse** sind Abläufe in der Programmlogik, die zwar vom Benutzer ausgelöst sein können, während deren Ausführung jedoch eine Interaktion zwischen Benutzer und System nicht notwendig und meist auch nicht erwünscht ist. Dieses Notationssymbol wird vor allem zur **Modellierung von wichtigen internen Algorithmen** des Systems verwendet.

  Anmerkung: innerhalb jedes Prozesses treten in der Regel sehr viele systeminterne Vorgänge auf. Mit Hilfe des in Abbildung 62 gezeigten Symbols werden jedoch nur solche Prozesse dokumentiert, welche **längere Systemprozesse** auslösen, z.B. einen bedeutenden Algorithmus. Andernfalls würde die Komplexität der Modelle steigen, was sich negativ auf die Lesbarkeit derselben auswirkt.
- **Alternative** ist ein Darstellungselement zur Verbindung von Aktivitäten oder Prozessen, von denen jeweils eine(r) beliebige(r) nach den Präferenzen des Benutzers ausgeführt werden kann. (Verknüpfung mit exklusivem Oder.)
- **Alternativen mit Bedingung** verknüpfen die Wahlmöglichkeit von alternativen Pfaden mit einer oder mehreren bestimmten Bedingungen.
- **Assertionen** modellieren Voraussetzungen, die wahr sein müssen, bevor ein Prozess oder eine Aktivität ausgeführt werden kann. Die Assertionen müssen so formuliert sein, dass sie mit „wahr" / „falsch" beantwortet werden können.
- Eine **Berechtigung** entspricht der **Befugnis zur Ausführung von Aktivitäten**, welche ein **Benutzer besitzen** und wahrnehmen kann. Berechtigungen sind im Rahmen der Benutzermodellierung auf Rollen (Klassen von Benutzern) zugeordnet.
- Eine **Datenbasis** steht für ein abstraktes Objekt in Form eines **dauerhaften, zentralen Speicherplatzes**, aus dem Daten gelesen oder in den Daten geschrieben werden können.
- Eine **Festplatte** symbolisiert einen dauerhaften, auf einem Einzelrechner lokal vorhandenen Datenspeicher, aus dem Daten gelesen oder in den Daten geschrieben werden können.
- **Start** und **Ende** markieren jeweils den Start- und Endpunkt eines Prozesses.
- **Verbindungen zwischen Prozessen** (durchgezogene Pfeile) bestimmen die Sequenz bei der Ausführung von Prozessen und Aktivitäten. Die Richtung des Pfeils gibt dabei die **Reihenfolge** an, in denen die Prozesse und Aktivitäten verpflichtend ausgeführt werden.

- Ein **Daten-Prozess-Konnektor** (durch einen gepunkteten Pfeil dargestellt) symbolisiert **Lese- oder Schreibzugriffe** eines Prozesses auf eine Datenbasis oder einen Datenspeicher. Die Pfeilspitzen geben dabei an, in welche Richtung Daten fließen. Möglichkeiten dabei sind (1) vom Prozess (oder der Aktivität) zur Datenbasis oder zum Datenspeicher, (2) von der Datenbasis oder dem Datenspeicher zum Prozess (oder zur Aktivität), (3) in beide Richtungen.
- **Rollen-Prozess-Konnektoren** (durch strichlierte Linien mit Punkten an beiden Enden der Linie dargestellt) schließlich werden für die Zuordnung von Prozessen zu Rollen (Benutzerklassen) verwendet.

6.2.2 Interaktionsprozesse

Abbildung 63 zeigt das **Modell für den Einstieg** in die Verwaltung des Contentpools. Die Contentpool-Verwaltung kann nur gestartet werden, wenn die Scholion WB+ Applikation (das Client-Frontend) geöffnet und das **Datenbanksystem gestartet** wurde. Andernfalls kann ein Datenaustausch zwischen den Client- und Serverkomponenten nicht stattfinden.

Nach dem Start der Contentpool-Verwaltung hat der Benutzer die Möglichkeit, verschiedene **Anwendungen zu starten** und ihre **Features zu nutzen**. Zur Auswahl stehen der **Dokument-Splitter** (siehe Abbildung 64: Modell des Dokument-Splitters, vgl. Abschnitt 6.2.2.1, Seite 174 ff.), das **Präsentationstool** (siehe Abbildung 77: Topic-Editor: Präsentations-Tool, vgl. Abschnitt 6.2.2.2.1, Seite 191 ff.[33]), der **Topic-Editor** oder Verwaltungstool (siehe Abbildung 76: Modell des Topic-Editors, vgl. Abschnitt 6.2.2.2, Seite 189 ff.), der **Topicmap-Manager** (siehe Abbildung 92: Modell des Topicmap-Managers, vgl. Abschnitt 6.2.2.3, Seite 206 ff.) sowie das **Hilfe-System** (siehe Abbildung 98: Modell des Contentpool-spezifischen Hilfe-Systems, vgl. Abschnitt 6.2.2.5, Seite 215 ff.).

Eine Sonderstellung nimmt die **Suchfunktion** ein. Diese ist eine **Querschnittsfunktion der Contentpool-Verwaltung**; d.h. die Suchfunktion kann von allen anderen Anwendungen aus gestartet und genutzt werden. Die Spezifikation der Suchfunktion kann aus Abbildung 97: Präsentations-Tool: Suche im Contentpool entnommen werden (vgl. auch Abschnitt 6.2.2.4 Suche im Contentpool, Seite 213 ff.).

Zu beachten ist, dass für jede der genannten Anwendungen der Benutzer über die **notwendige Berechtigung** verfügen muss. Diese kann ihm durch einen Administrator des Scholion WB+ Systems zugewiesen werden. Eine Übersicht über mögliche Berechtigungen für die Contentpool-Verwaltung findet sich in Abschnitt 6.2.4 (Berechtigungsbaum für die Contentpool-Verwaltung, Seite 221 ff.).

Die Interaktion zwischen dem System und dem Benutzer endet, sobald sich der Benutzer entscheidet, die Contentpool-Verwaltung zu verlassen.

[33] Anmerkung: das Präsentationstool wurde logisch dem Topic Editor untergeordnet, da es für die Features dieses Tools eine ganz wesentliche Funktionalität bereitstellt. Es ist jedoch auch möglich, nach dem Start der Content-Pool Verwaltung sofort das Präsentationstool zu starten. Dies ist der Grund dafür, wieso das Präsentationstool auf der obersten Ebene der Prozessmodelle aufscheint.

In den folgenden Abschnitten werden die Prozessmodelle zu den einzelnen, am Beginn dieses Abschnitts genannten Werkzeugen (Tools) der Contentpool-Verwaltung präsentiert:

- 6.2.2.1 Dokument-Splitter .... Seite 174
- 6.2.2.2 Topic-Editor .... Seite 189
- 6.2.2.3 Topicmap-Manager .... Seite 206
- 6.2.2.4 Suche im Contentpool .... Seite 213
- 6.2.2.5 Hilfe-System .... Seite 215

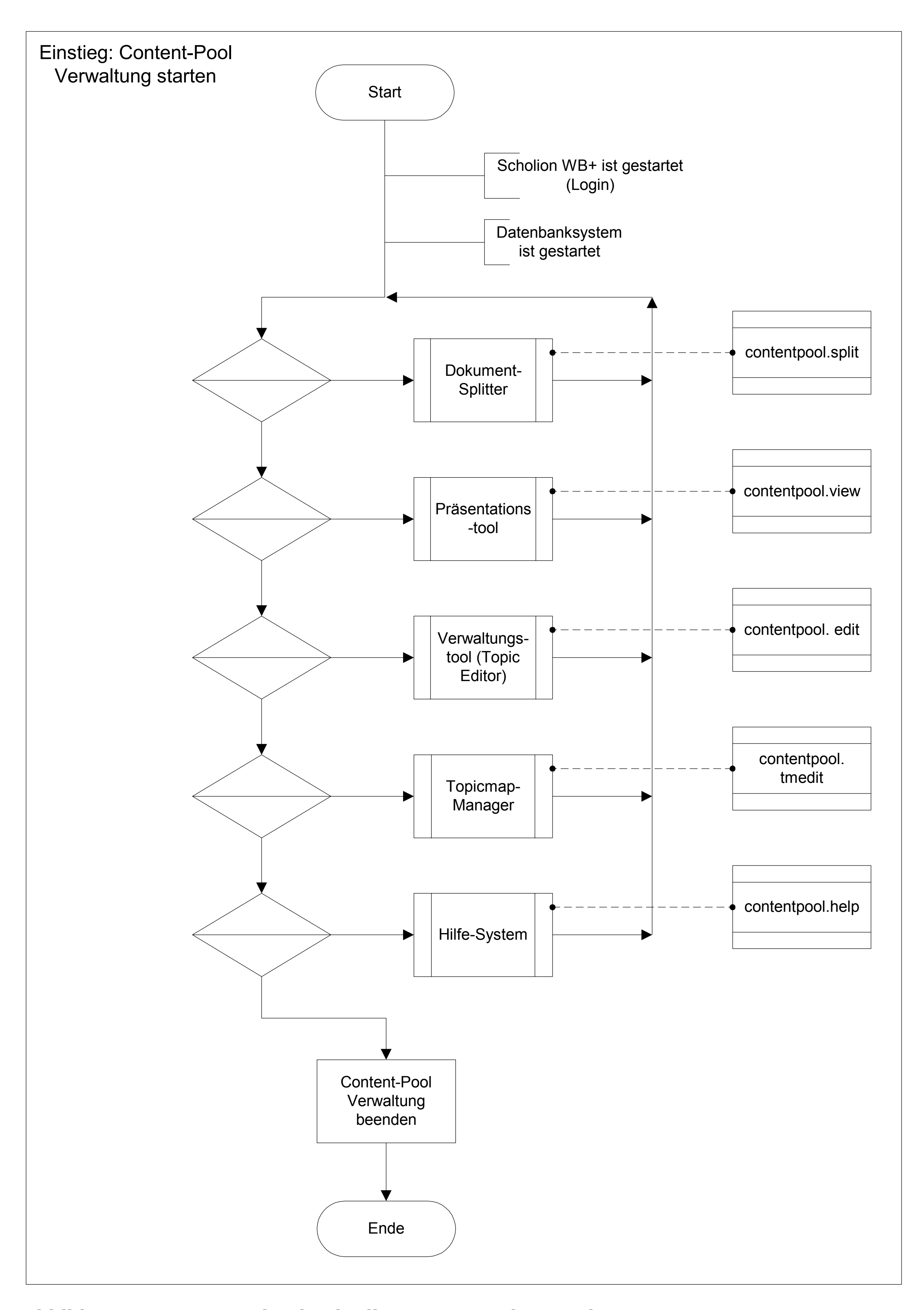


Abbildung 63: Prozess Einstieg in die Contentpool-Verwaltung

6.2.2.1 Dokument-Splitter

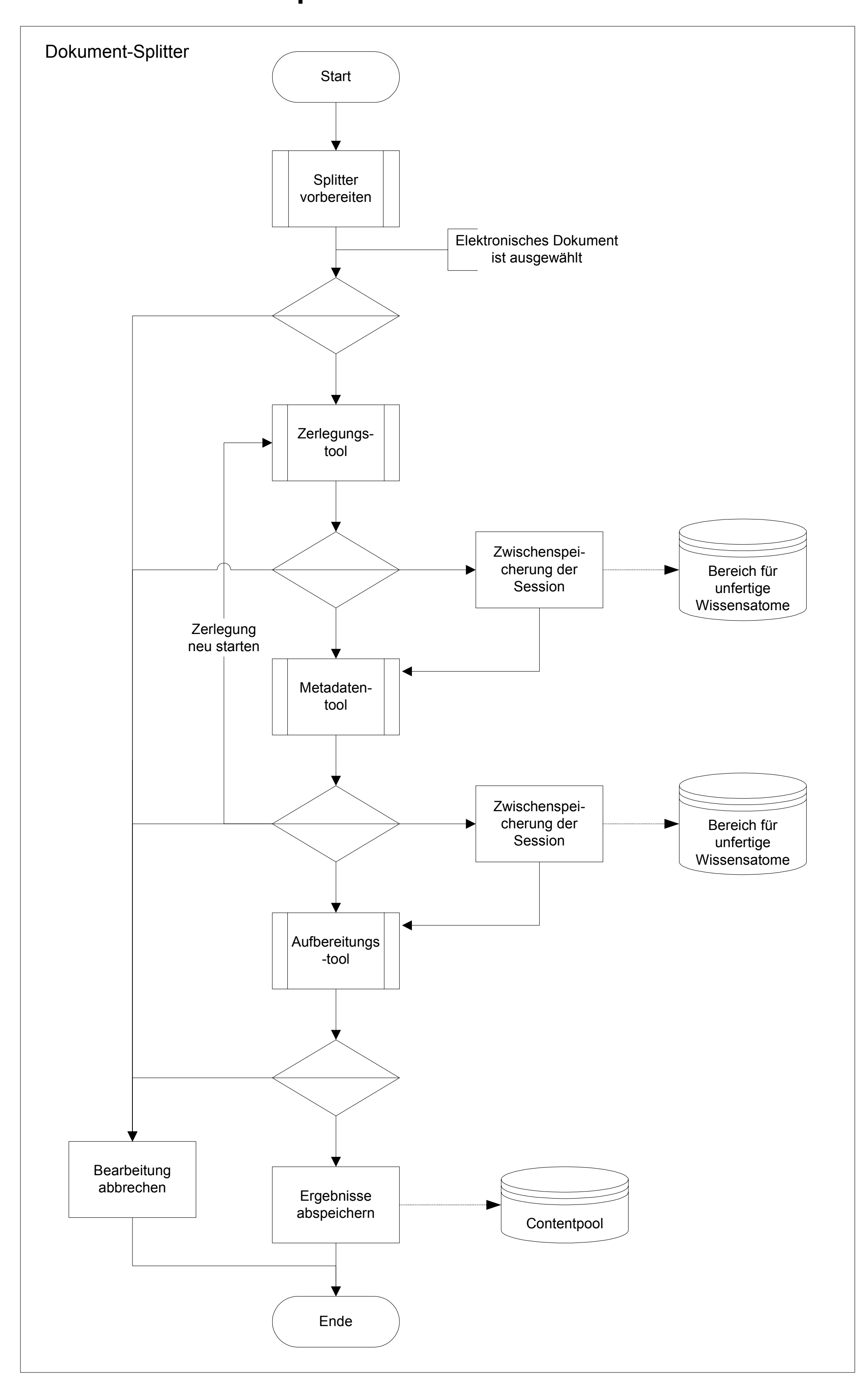


Abbildung 64: Modell des Dokument-Splitters

Beschreibung: der Dokument-Splitter ist jenes Tool, mit dessen Hilfe neue elektronische Dokumente in den Contentpool eingefügt werden können. Bevor jedoch die automatische Zerlegung gestartet werden kann, hat der Benutzer diverse **Parameter des Algorithmus** zu konfigurieren. Anschließend werden (exakt in der angegebenen Reihenfolge) das **Zerlegungs-Tool**, das **Metadaten-Tool** und das **Aufbereitungs-Tool** gestartet. Nach jedem dieser Schritte hat der Benutzer die Möglichkeit, Zwischenergebnisse zu speichern, um die Bearbeitung unterbrechen und zu einem späteren Zeitpunkt wieder fortsetzen zu können.

Splitter-Algorithmus vorbereiten: der erste Schritt besteht darin, ein elektronisches **Dokument** zu öffnen. Das Dokument muss einem der unterstützten Dateiformate entsprechen[34]. Anschließend müssen die **allgemeinen Parameter** konfiguriert werden. Dazu gehört zuerst die Definition des Fachbereichs, dem das zerlegte Dokument zugeordnet werden soll. Dies beeinflusst, in welchem semantischen Netz die zu erzeugenden Wissensatome abgelegt werden und hat so wichtige Auswirkungen auf den Wert und die Wiederverwendbarkeit der erzeugten Wissensatome. Optional kann zu diesem Zeitpunkt der **Topicmap-Manager** gestartet werden, der die Definition neuer Fachgebiete erlaubt (vgl. Abschnitt 6.2.2.3, Seite 206 ff.). Schließlich ist noch der **Zerlegungsbereich** zu definieren, d.h. ab welcher Stelle im elektronischen Dokument die Zerlegung zu beginnen hat und an welcher Stelle sie zu beenden ist. Auch das gesamte Dokument kann dabei ausgewählt werden. Der definierte Zerlegungsbereich wird als *ein Buch* im logischen Sinn betrachtet, d.h. für diesen Bereich ist sicherzustellen, dass ein Autor oder ein Publikationstitel eindeutig definierbar sind. Für die Brauchbarkeit der Metadaten ist dies von entscheidender Bedeutung (vgl. dazu auch Abschnitt 6.2.2.1.3 Metadaten-Tool, Seite 182 ff. und Abschnitt 6.2.2.1.4 Aufbereitungs-Tool, Seite 185 ff.). Sammelbände, Herausgeberwerke und Ähnliches gelten demnach für die Contentpool-Verwaltung als mehrere Bücher, so dass folglich auch der Dokument-Splitter in solchen Fällen mehrmals gestartet werden muss.

Zerlegungs-Tool starten: Bedingung für den Start des Zerlegungs-Tool ist, dass zuvor ein elektronisches Dokument entsprechend der obigen Beschreibung geöffnet wurde. Anschließend sind die **Parameter für die Zerlegung** zu definieren. Eine **Default-Konfiguration** wird vom System vorgeschlagen. Dem Benutzer wird zudem die Möglichkeit geboten, eine **Konfigurationsdatei** zu öffnen, in welcher eine bestimmte Konfiguration der Parameter gespeichert ist. Festzulegen sind in jedem Fall die ge-

[34] Anmerkung: zu unterstützende Dateiformate werden im Verlauf der Implementierungsphase des Scholion WB+ Content-Pools noch festgelegt werden. In jedem Fall wird jedoch das Dateiformat PDF unterstützt werden. Die Unterstützung weiterer Dateiformate wird durch die Implementierung eines modulbasierten Importfilter-Systems realisiert werden.

wünschte Größe der Wissensatome (wobei Maßeinheiten wie eine Zeile, ein Absatz, ein Kapitel usw. herangezogen werden); Formatierungsregeln (Formatvorlagen) für Kapitelüberschriften, Objektbeschriftungen und Ähnliches; die Position eines eventuell vorhandenen Inhalts-, Schlagwort- und Literaturverzeichnisses. Die zu konfigurierenden Parameter beeinflussen wesentlich die Brauchbarkeit der automatisch erzeugten Wissensatome! So wie die gesamte Zerlegung sollte die Konfiguration daher nur von einem Experten auf dem entsprechenden Fachgebiet vorgenommen und die Parameter sorgfältig gewählt werden. Vorgenommene Konfigurationen können schließlich vom Benutzer auf dem Scholion WB+ Server in einer **Datei gespeichert** werden, wobei auf die Eingabe eines gültigen Dateinamens zu achten ist.

Metadaten-Tool starten: nach der Durchführung der automatischen Zerlegung an Hand der getroffenen Konfiguration der Zerlegungsparameter (vgl. die obige Beschreibung) wird für die vorläufigen Ergebnisse (d.h. für die Wissensatome in ihrer vorläufigen Form) eine **automatische Extraktion** von Metadaten gestartet. Dabei wird aus dem elektronischen Dokument wichtige Information über den Inhalt extrahiert, welche später für die effiziente Auffindung der Wissensatome unverzichtbar ist. Dem Benutzer obliegt auch hier die **Definition von Parametern**, welche dem Tool das Auffinden der Metadaten im Dokument erleichtern oder erst ermöglichen. Eine **Default-Konfiguration** wird vom System vorgeschlagen. Dem Benutzer wird zudem die Möglichkeit geboten, eine **Konfigurationsdatei** zu öffnen, in welcher eine bestimmte Konfiguration der Parameter gespeichert ist. Festzulegen sind der Namen des Autors (bzw. Herausgebers); der Titel der Publikation; das Erscheinungsjahr und der Verlag, von dem das Werk publiziert wird; die Definition von Formatierungsregeln (Formatvorlage) für Querverweise im Dokument. Vorgenommene Konfigurationen der Parameter können vom Benutzer auf dem Scholion WB+ Server in einer **Datei gespeichert** werden, wobei auf die Auswahl eines gültigen Dateinamens zu achten ist.

Aufbereitungs-Tool starten: nach der automatischen Durchführung der Zerlegung in Wissensatome und die Aufbereitung der Wissensatome mit Metadaten (siehe die obigen Beschreibungen der beiden Schritte) liegt ein **vorläufiges Zerlegungsergebnis** in Form von Wissensatomen vor. Ohne eine manuelle Nachbearbeitung sind diese Wissensatome jedoch noch nicht geeignet, in den Contentpool integriert zu werden. Für diesen Zweck kommt das Aufbereitungs-Tool ins Spiel. Vom Benutzer müssen dabei die Wissensatome „richtig", d.h. semantisch sinnvoll, **geteilt** werden, indem zu große Wissensatome gesplittet und zu kleine Wissensatome wieder zusammen gefügt werden können. Die generierten **Metadaten** müssen manuell überarbeitet werden. Jene Teile, die nicht automatisch generierbar sind, müssen zudem ergänzt werden. Schließlich müssen auch noch die **semantischen Relationen** zwischen den Wissensatomen definiert werden. Wenn zwei Wissensatome zueinander in semantischer

Relation stehen, so sind vom Benutzer (1) die beiden beteiligten (verknüpften) Wissensatome, (2) die Richtung der Verknüpfung sowie (3) die Bedeutung der Verknüpfung zu definieren. Bei der Aufbereitung von Zerlegungsergebnissen ist in der Regel mit einem **sehr hohen Zeitaufwand** zu rechnen (vor allem für die Erstellung der semantischen Verknüpfungen), weshalb der Benutzer die Möglichkeit zur Zwischenspeicherung der vorläufigen Ergebnisse und späteren Weiterbearbeitung erhält.

In der Folge werden die Prozessmodelle für die oben genannten und beschriebenen Tools präsentiert.

- 6.2.2.1.1 Vorbereiten des Dokument-Splitters .... Seite 178
- 6.2.2.1.2 Zerlegungs-Tool .... Seite 179
- 6.2.2.1.3 Metadaten-Tool .... Seite 182
- 6.2.2.1.4 Aufbereitungs-Tool .... Seite 185

6.2.2.1.1 Vorbereiten des Dokument-Splitters

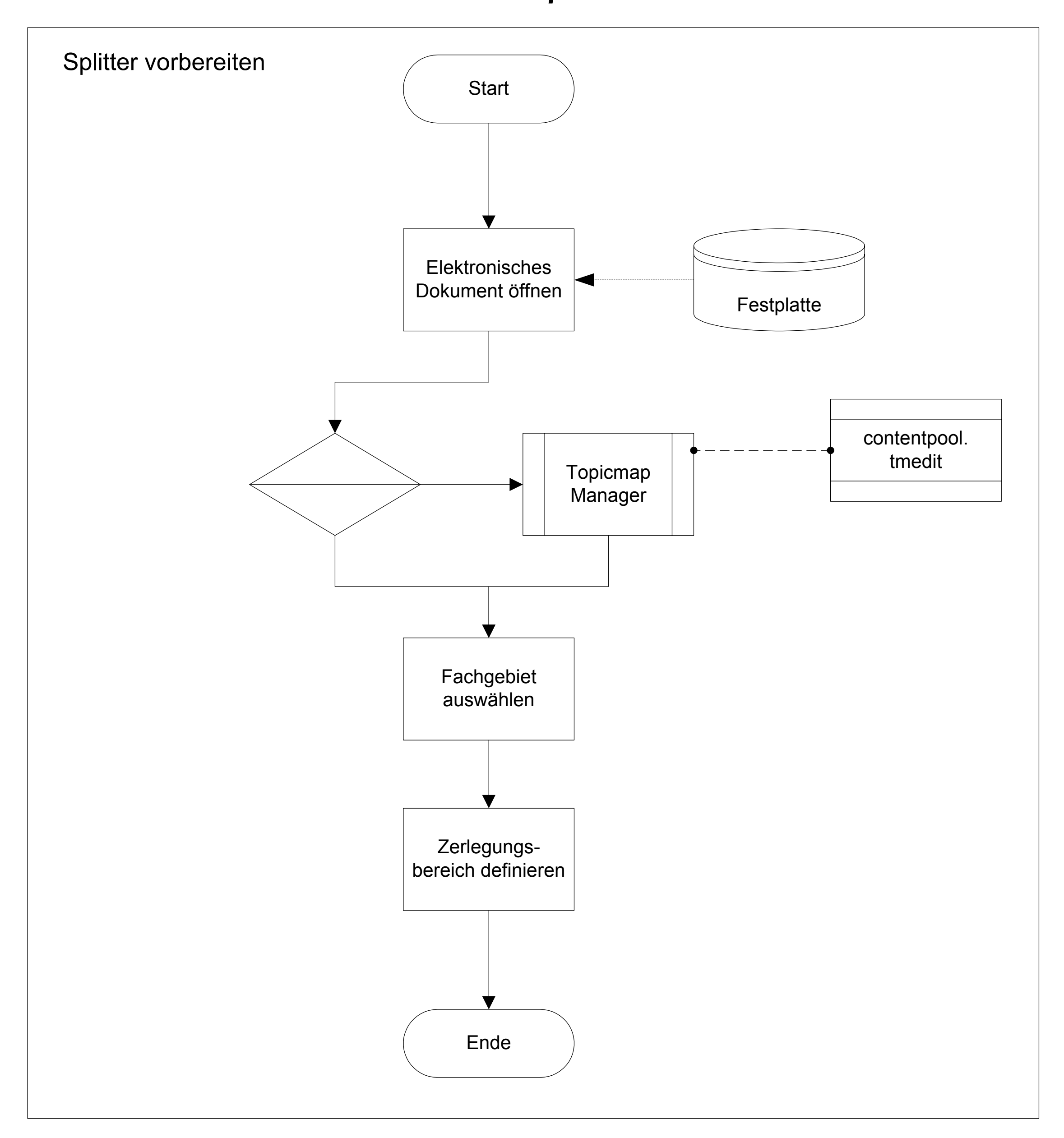


Abbildung 65: Dokument-Splitter: Splitter vorbereiten

6.2.2.1.2 Zerlegungs-Tool

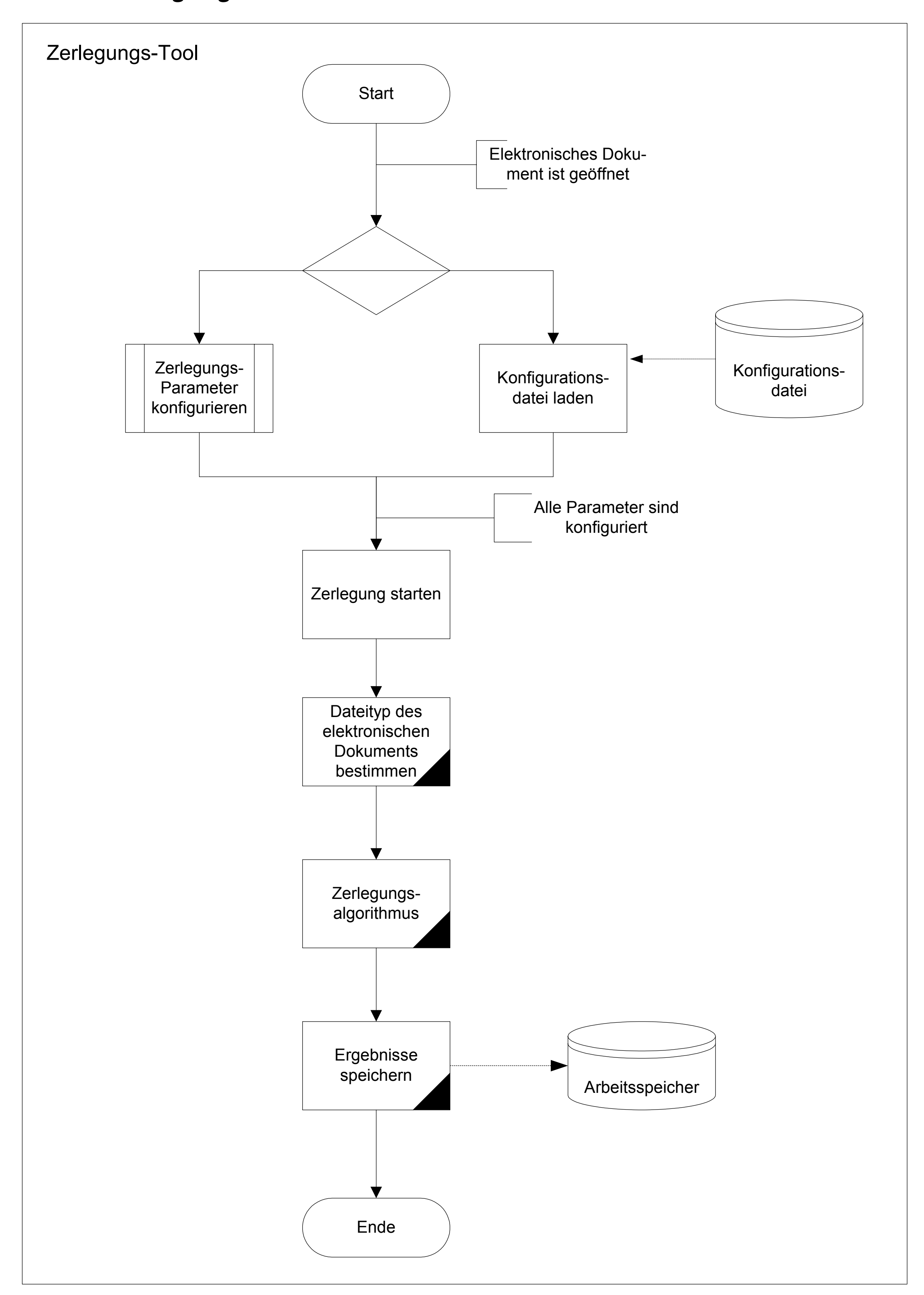


Abbildung 66: Dokument-Splitter: Zerlegungs-Tool

6.2.2.1.2.1 Zerlegungsparameter konfigurieren

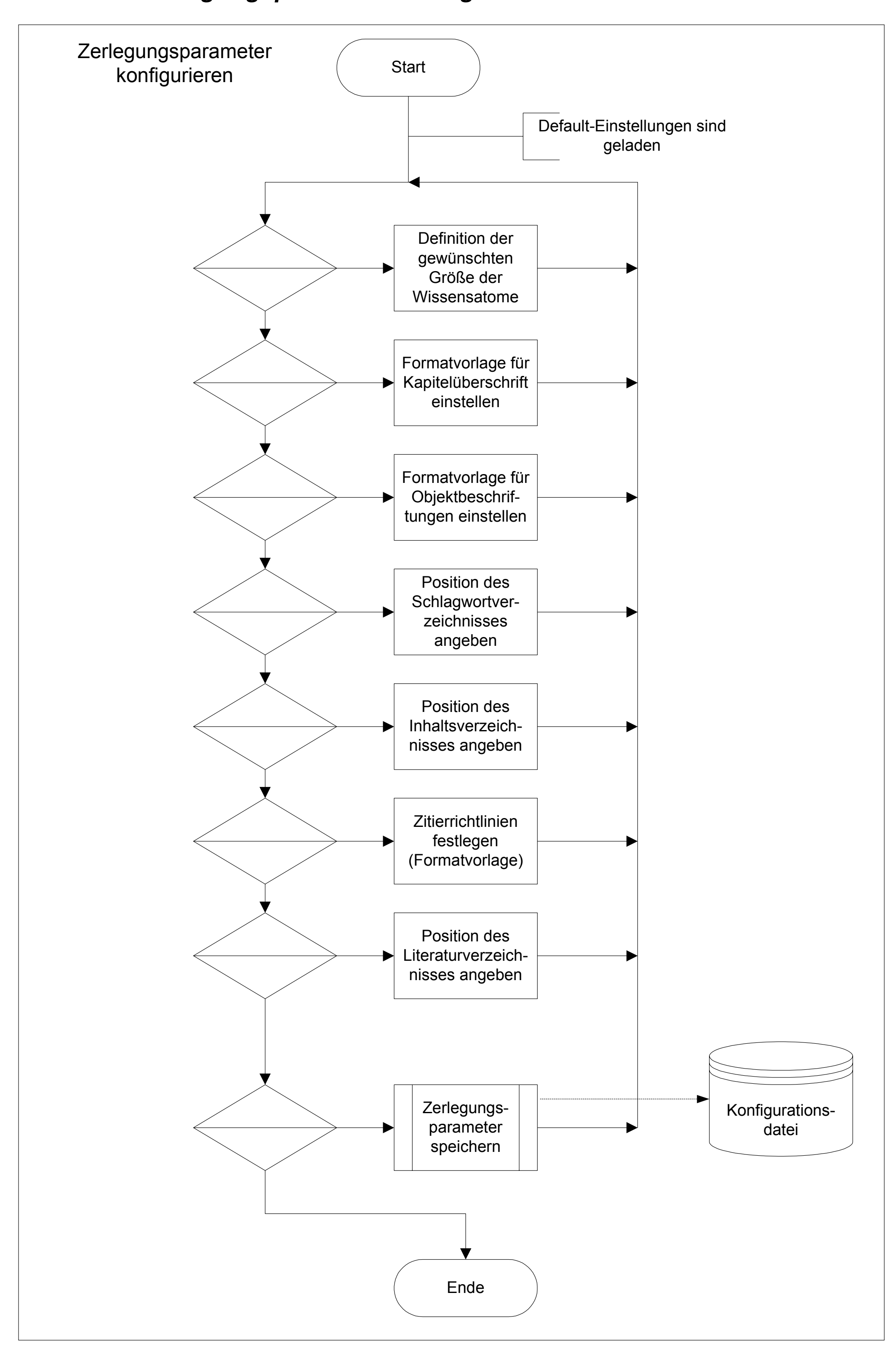


Abbildung 67: Zerlegungs-Tool: Zerlegungsparameter konfigurieren

6.2.2.1.2.2 Zerlegungsparameter speichern

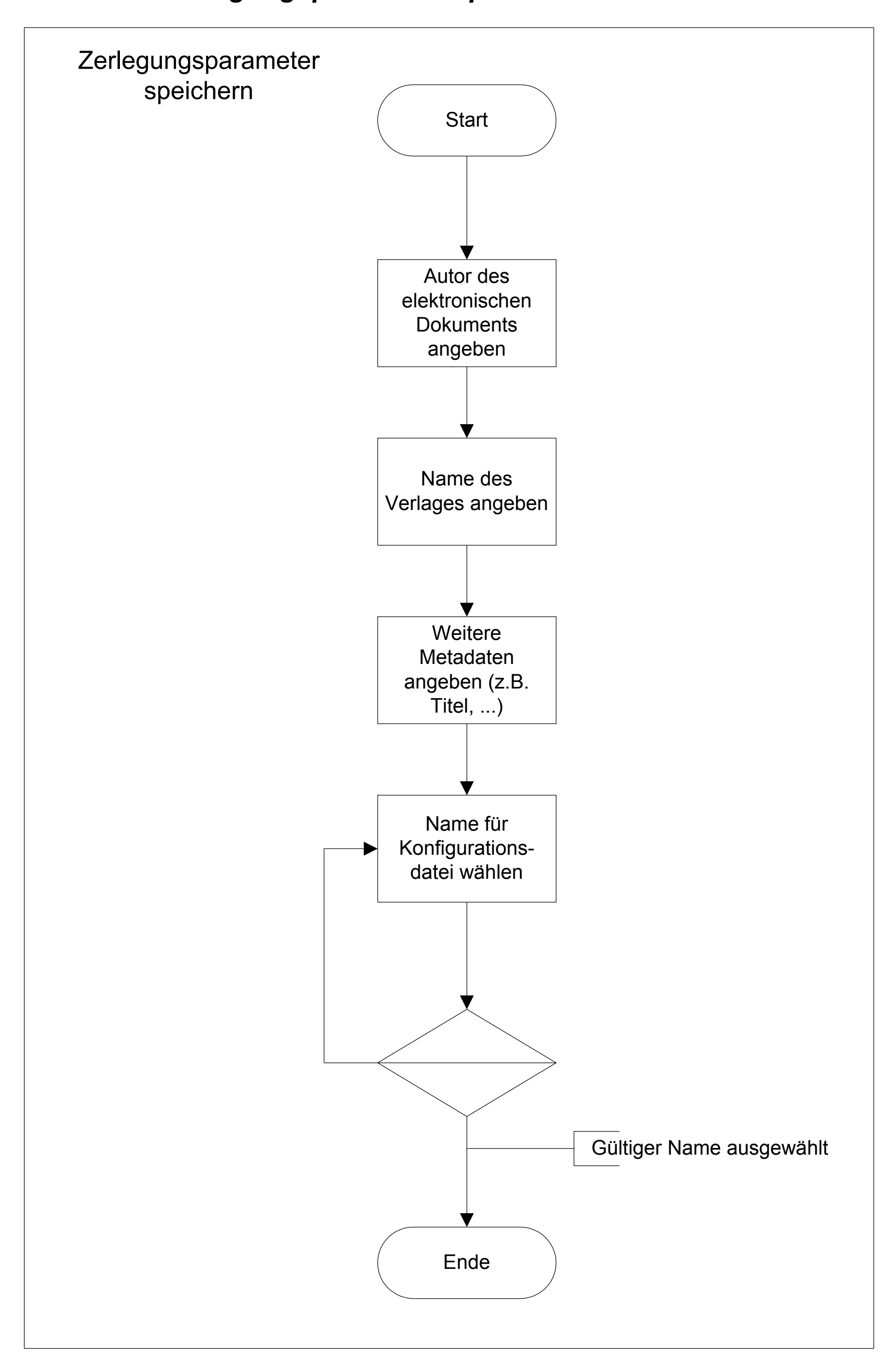


Abbildung 68: Zerlegungs-Tool: Zerlegungsparameter speichern

6.2.2.1.3 Metadaten-Tool

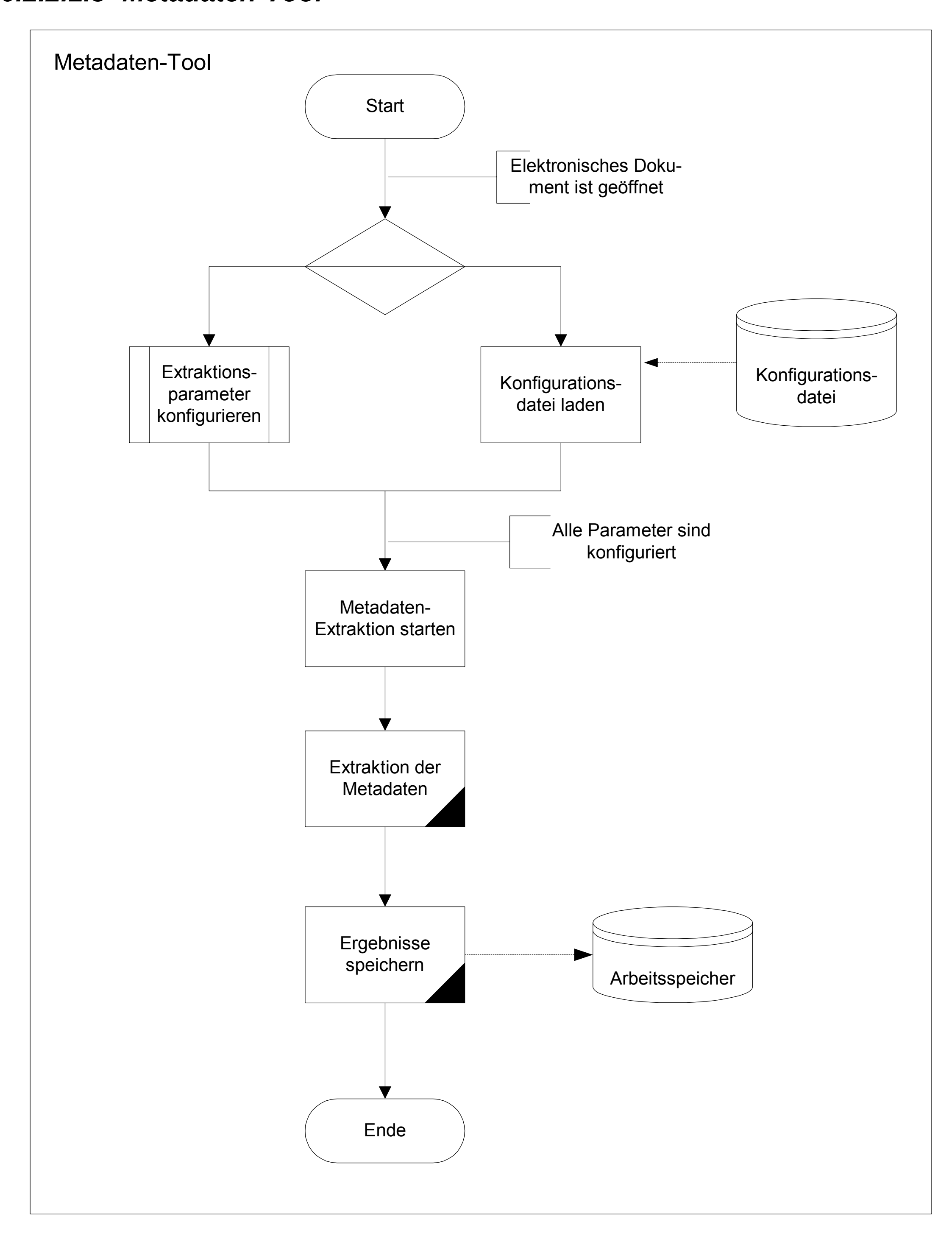


Abbildung 69: Dokument-Splitter: Metadaten-Tool

6.2.2.1.3.1 Extraktionsparameter konfigurieren

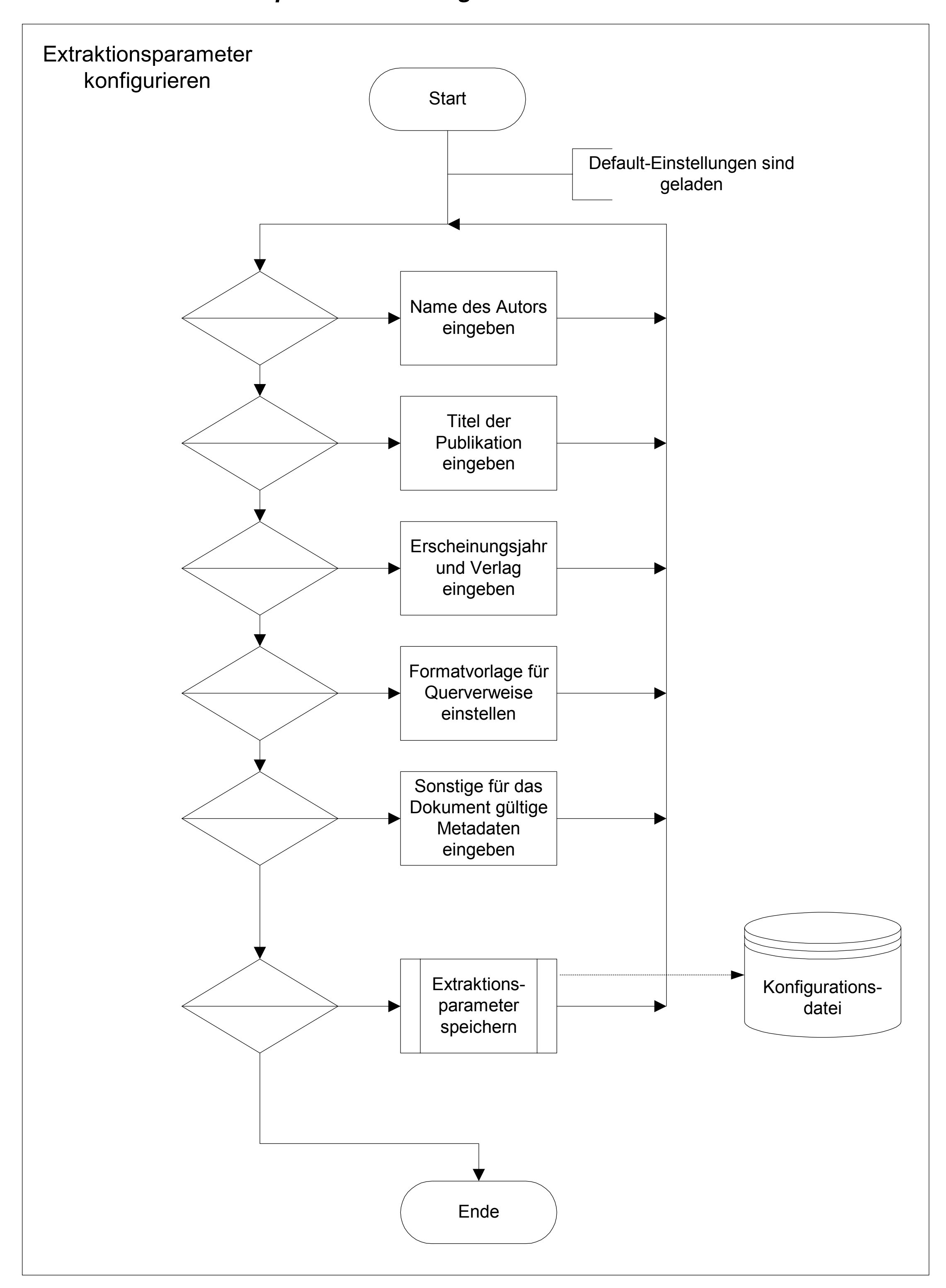


Abbildung 70: Metadaten-Tool: Extraktionsparameter konfigurieren

6.2.2.1.3.2 Extraktionsparameter speichern

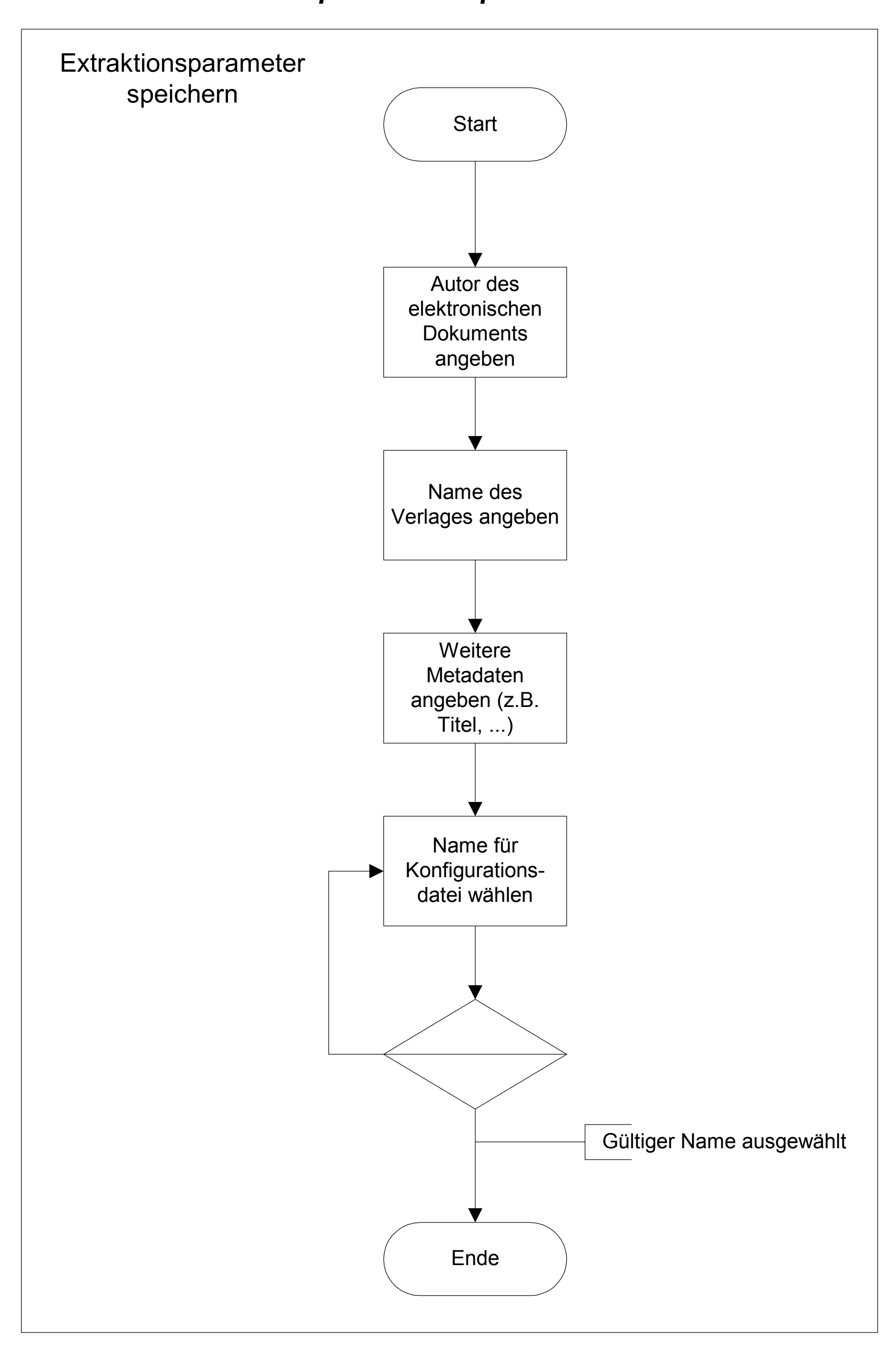


Abbildung 71: Metadaten-Tool: Extraktionsparameter speichern

6.2.2.1.4 Aufbereitungs-Tool

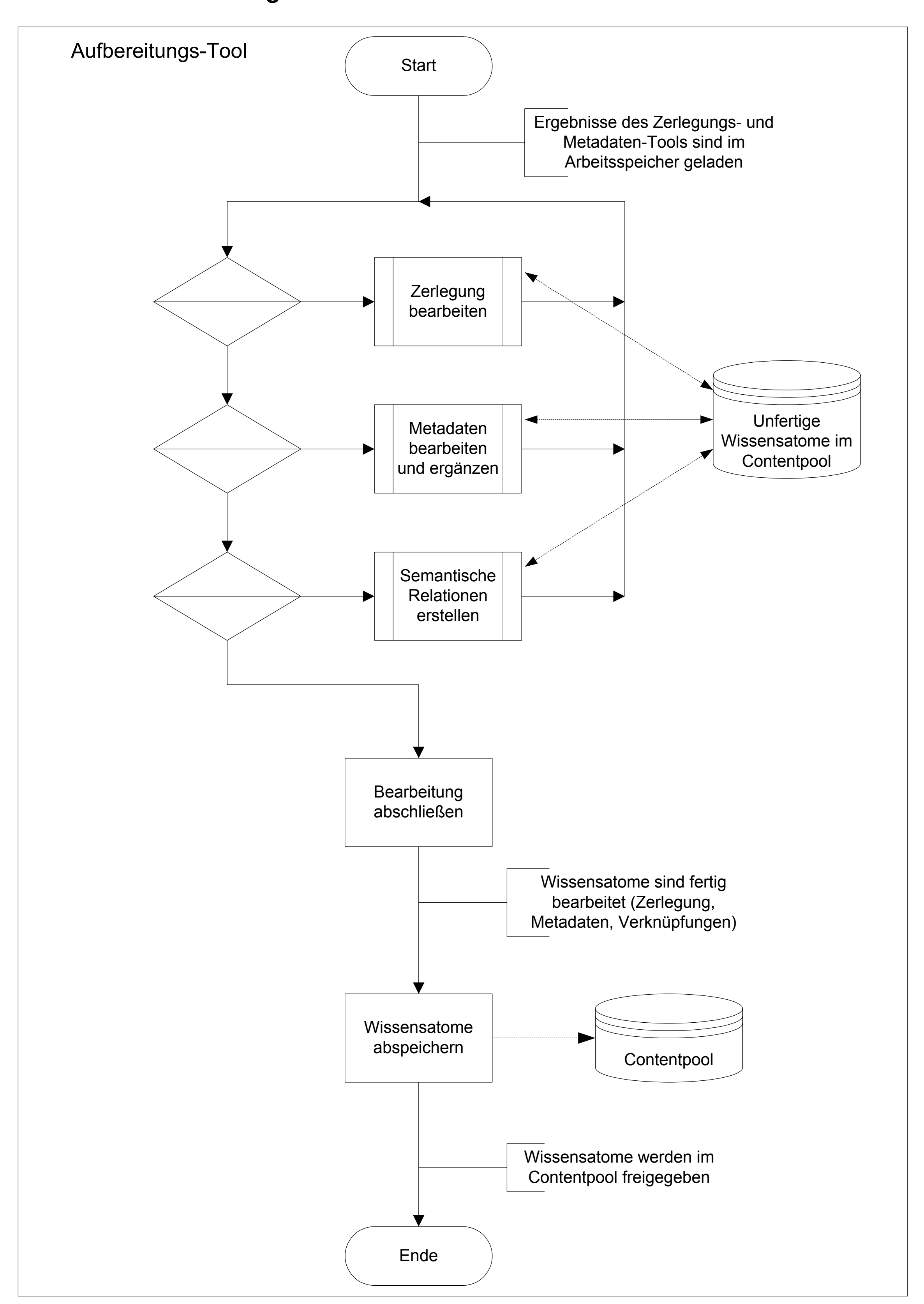


Abbildung 72: Dokument-Splitter: Aufbereitungs-Tool

6.2.2.1.4.1 Zerlegung bearbeiten

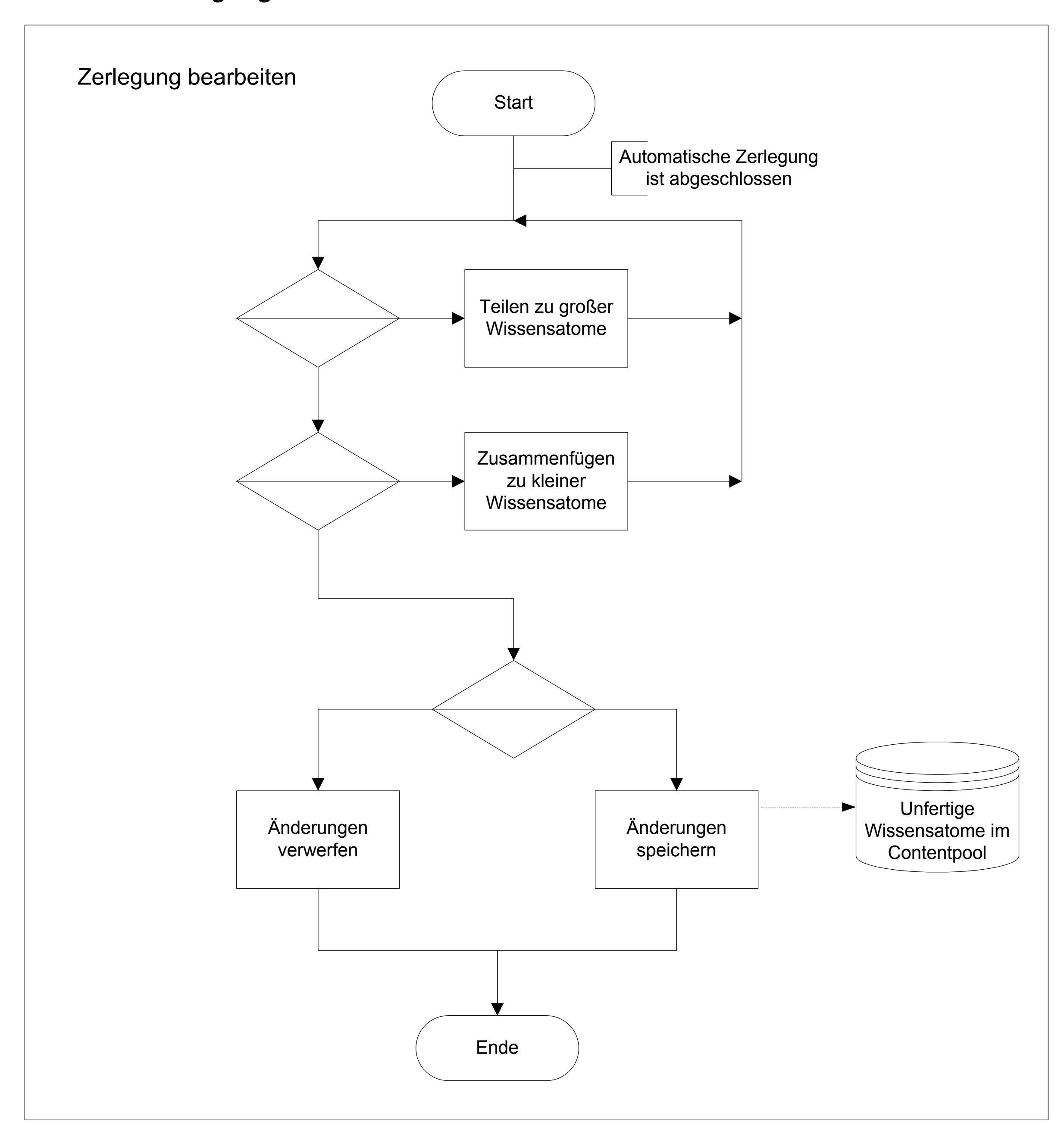


Abbildung 73: Aufbereitungs-Tool: Zerlegung bearbeiten

6.2.2.1.4.2 Metadaten bearbeiten und ergänzen

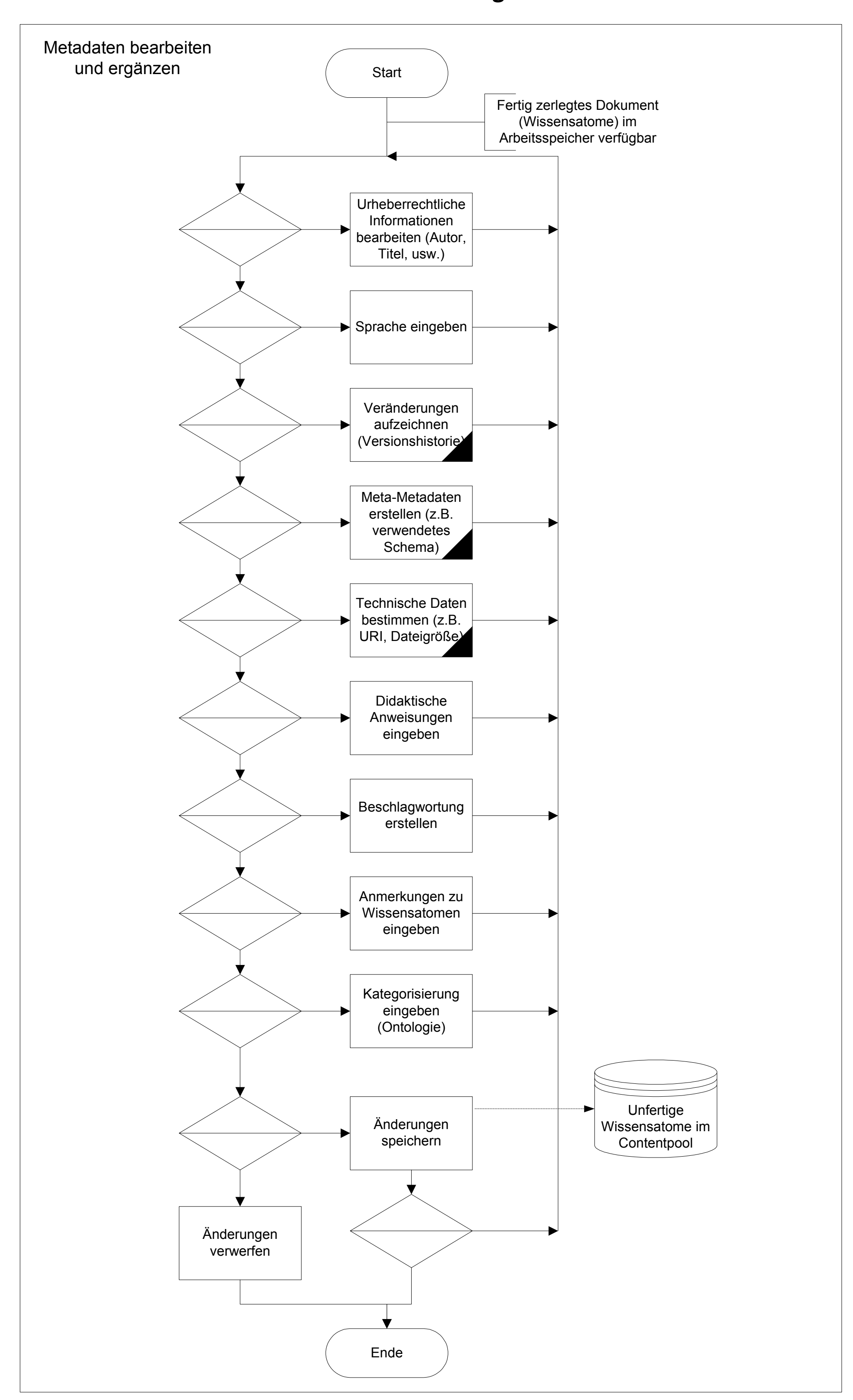


Abbildung 74: Aufbereitungs-Tool: Metadaten bearbeiten und ergänzen

6.2.2.1.4.3 Semantische Relationen erstellen

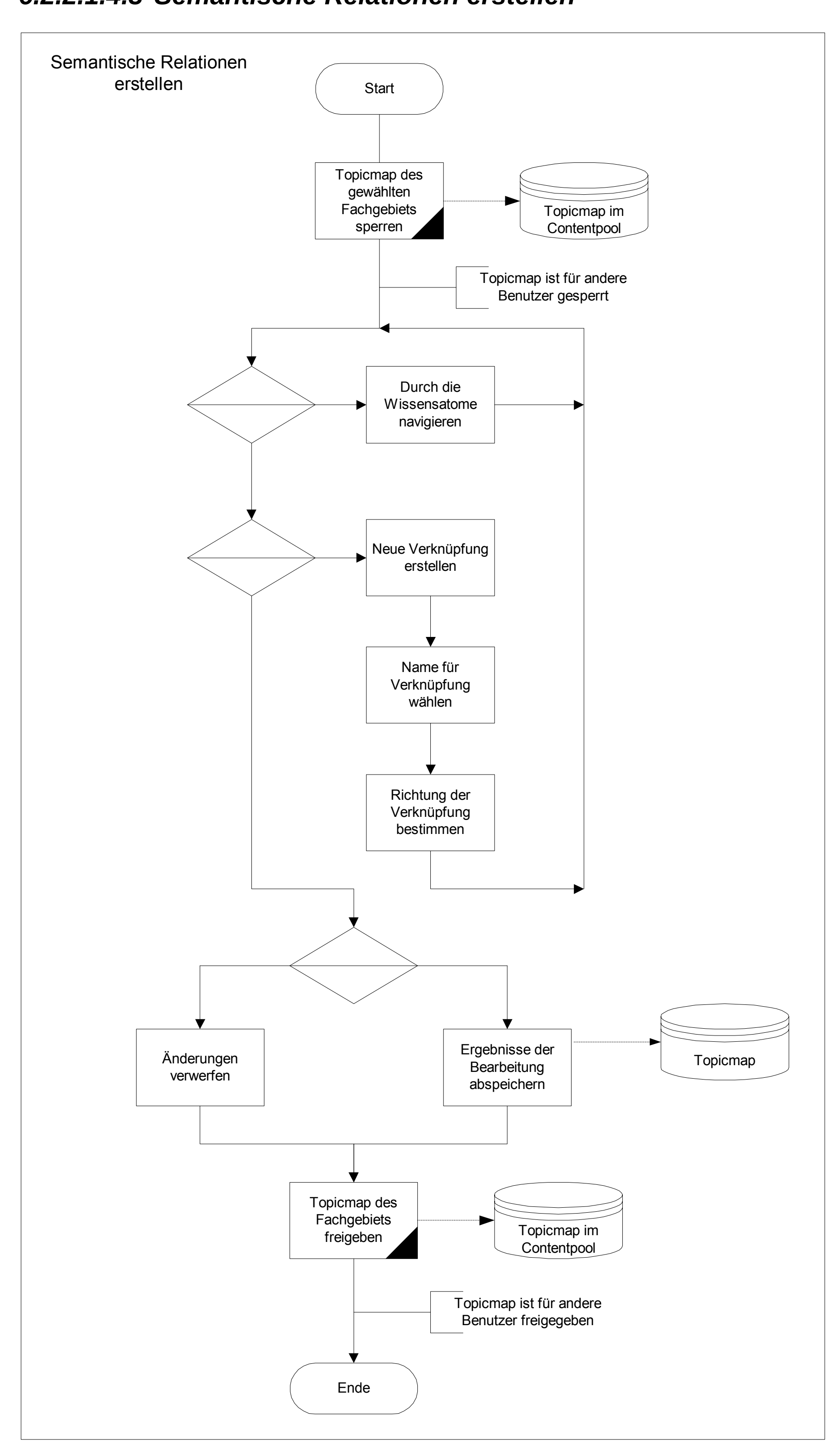


Abbildung 75: Aufbereitungs-Tool: semantische Relationen erstellen

6.2.2.2 Topic-Editor

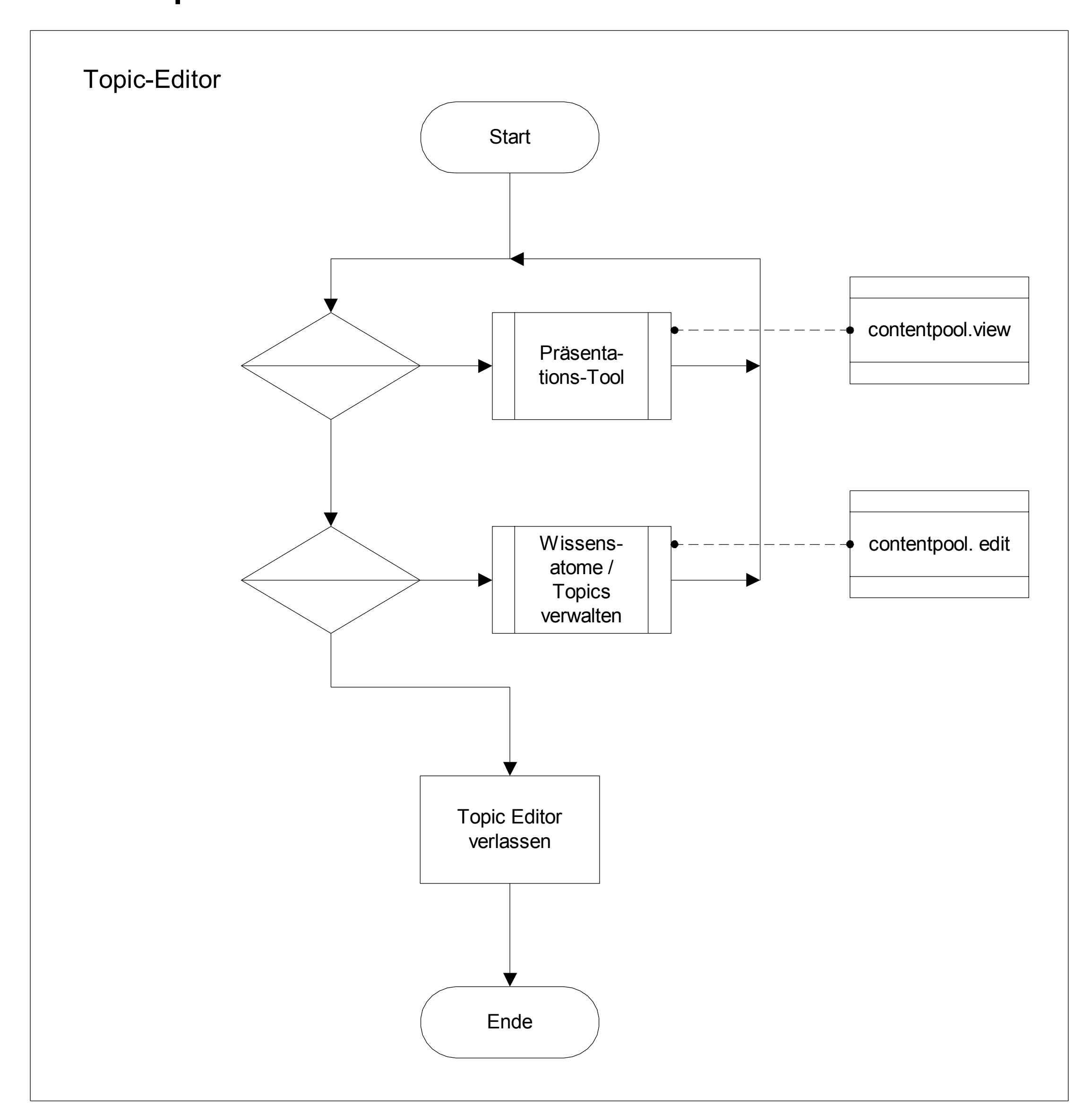


Abbildung 76: Modell des Topic-Editors

Beschreibung: Der Topic-Editor ist als Werkzeug für die **Administration** des Contentpools konzipiert. D.h. er bietet Funktionalität, um Wissensatome **hinzufügen, bearbeiten und löschen** zu können. Dies beinhaltet auch die Verwaltung der semantischen Netze im Contentpool. Konkret werden folgende Features unterstützt: Präsentationstool starten, Wissensatome verwalten. Alle zur Administration benötigten Funktionen sind in diesen beiden Werkzeugen beinhaltet.

Präsentationstool starten: das Präsentationstool wird für die effiziente **Navigation** im semantischen Netz und für die **Betrachtung** von Wissensatomen benötigt. Der Benutzer erhält die Möglichkeit, sowohl entsprechend der **ursprünglichen Struktur** der Kapitel aus den zerlegten Büchern als auch entlang der **semantischen Verknüpfungen** zu navigieren. Bei-

den Darstellungsweisen gemeinsam ist das Feature, Wissensatome für den Export in ein **persönliches Nachschlagewerk** auswählen und anschließend den Exportvorgang starten zu können. Im Präsentationstool kann eine **Suche** durchgeführt (vgl. auch Abschnitt 6.2.2.4, Seite 213 ff.) und die gefundenen Wissensatome anschließend betrachtet werden. Beim Anzeigen (Betrachten) der Wissensatome kann der Benutzer den Inhalt oder die Metadaten einsehen. Zudem können die Namen von semantischen Verknüpfungen betrachtet werden, so dass die Semantik der Wissensatome leicht erfassbar ist.

Wissensatome / Topics verwalten: für die Pflege des Contentpools werden Features (1) zum Erstellen neuer Wissensatome bzw. Topics, (2) zum Editieren und (3) zum Löschen bestehender Wissensatome bzw. Topics angeboten. Beim **Erstellen** eines neuen Wissensatoms kann der Inhalt entweder manuell eingegeben oder aber der Pfad (der URI) einer digitalen Ressource angegeben werden. Die **Metadaten** zum Wissensatom müssen vollständig manuell eingegeben sowie auch die **semantischen Verknüpfungen** zu anderen Wissensatomen erstellt werden. Die beiden letztgenannten Schritte ähneln in ihrer Durchführung stark dem Vorgehen beim Aufbereitungs-Tool des Dokument-Splitters (vgl. Abschnitt 6.2.2.1.4, Seite 185). Die selben Schritte mit Ausnahme der Eingabe des Inhalts (bzw. des Pfads der digitalen Ressource) sind beim **Editieren** eines Wissensatoms durchzuführen. Anstatt der Eingabe eines neuen Inhalts ist in diesem Fall die Bearbeitung eines bestehenden Inhalts vorgesehen. Soll ein Wissensatom **gelöscht** werden, braucht dieses lediglich ausgewählt und der Befehl zum Löschen gegeben werden.

Für das Erstellen eines **neuen Topics** müssen alle **Inhalte** des Topics und seine **Verknüpfungen** zu anderen Topics spezifiziert werden. Die Aufgaben überschneiden sich dabei teilweise mit denen bei der Erstellung von semantischen Verknüpfungen für Wissensatome. Beim **Editieren** eines Topics müssen die selben Schritte wie bei der Erstellung vollzogen werden. Statt Eingabe eines neuen wird jedoch der **bestehende Inhalt** eines Topics editiert. Für das **Löschen** eines Topics braucht es lediglich ausgewählt und der Befehl zum Löschen gegeben werden.

Alle Änderungen, die in der Wissensatom-Verwaltung vorgenommen werden, können aus Sicherheitsgründen **abgebrochen oder rückgängig** gemacht werden. Erst wenn die Wissensatom-Verwaltung beendet wird, werden sämtliche Änderungen unwiderruflich in den Contentpool übernommen. Daher ist beim Umgang mit diesem Werkzeug eine sorgfältige Vorgehensweise geboten!

In der Folge werden die Prozessmodelle für die oben genannten und beschriebenen Tools und Features präsentiert. Dabei wird die Verwaltung des Contentpools in zwei Teilen präsentiert, wobei nach den zu verwaltenden Objekten (Wissensatome oder Topics im Contentpool) unterschieden wird:

6.2.2.2.1 Präsentations-Tool

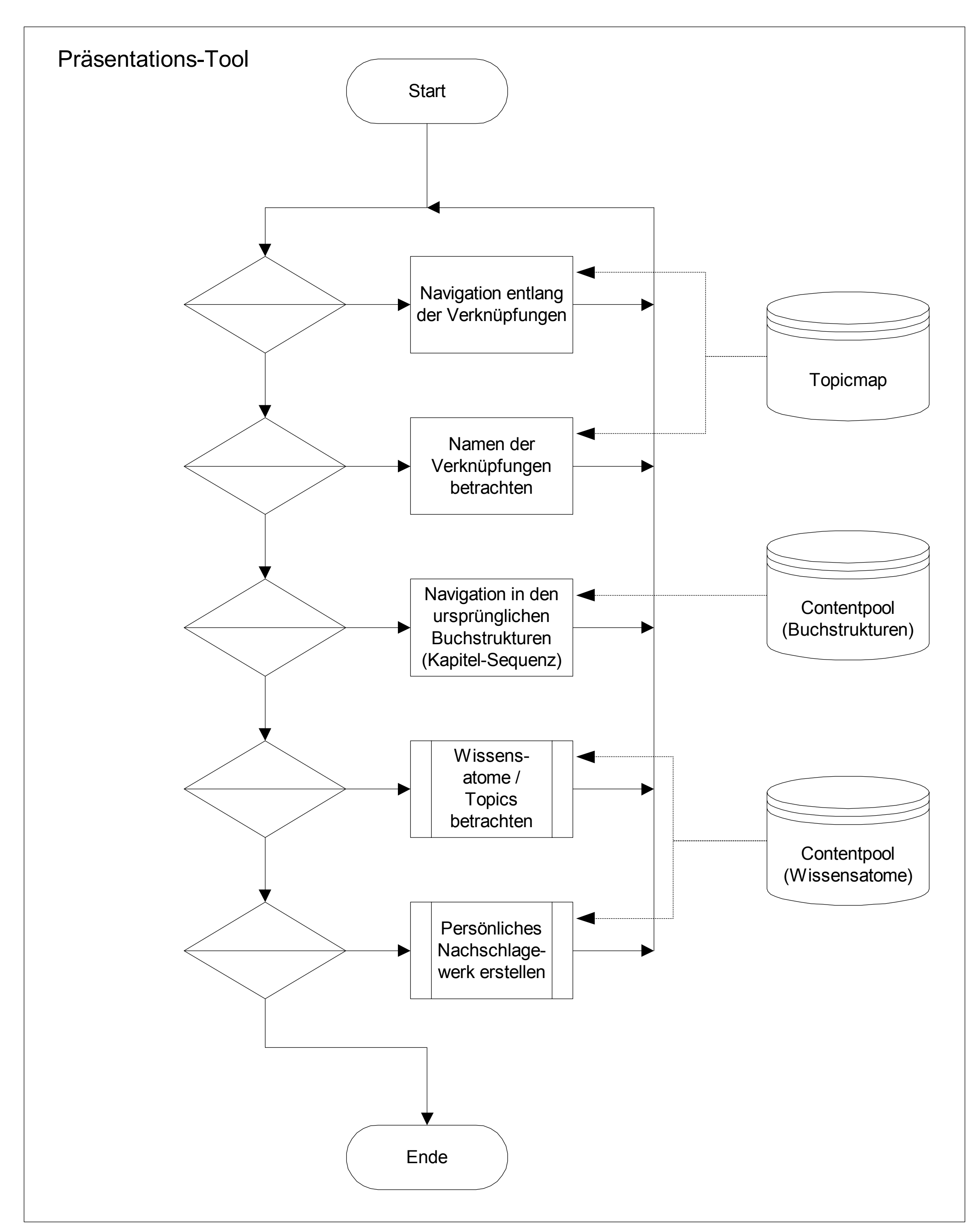


Abbildung 77: Topic-Editor: Präsentations-Tool

Hinweis: Der Schritt „Wissensatome / Topics betrachten" in Abbildung 77 wird in zwei Prozessmodelle verfeinert: „Wissensatome betrachten" (Abbildung 78) und „Topic be-

trachten" (Abbildung 79). Auf diese Weise werden einerseits die Unterschiede und andererseits die Zusammenhänge zwischen Wissensatomen und Topics besser verständlich.

6.2.2.2.1.1 Wissensatome betrachten

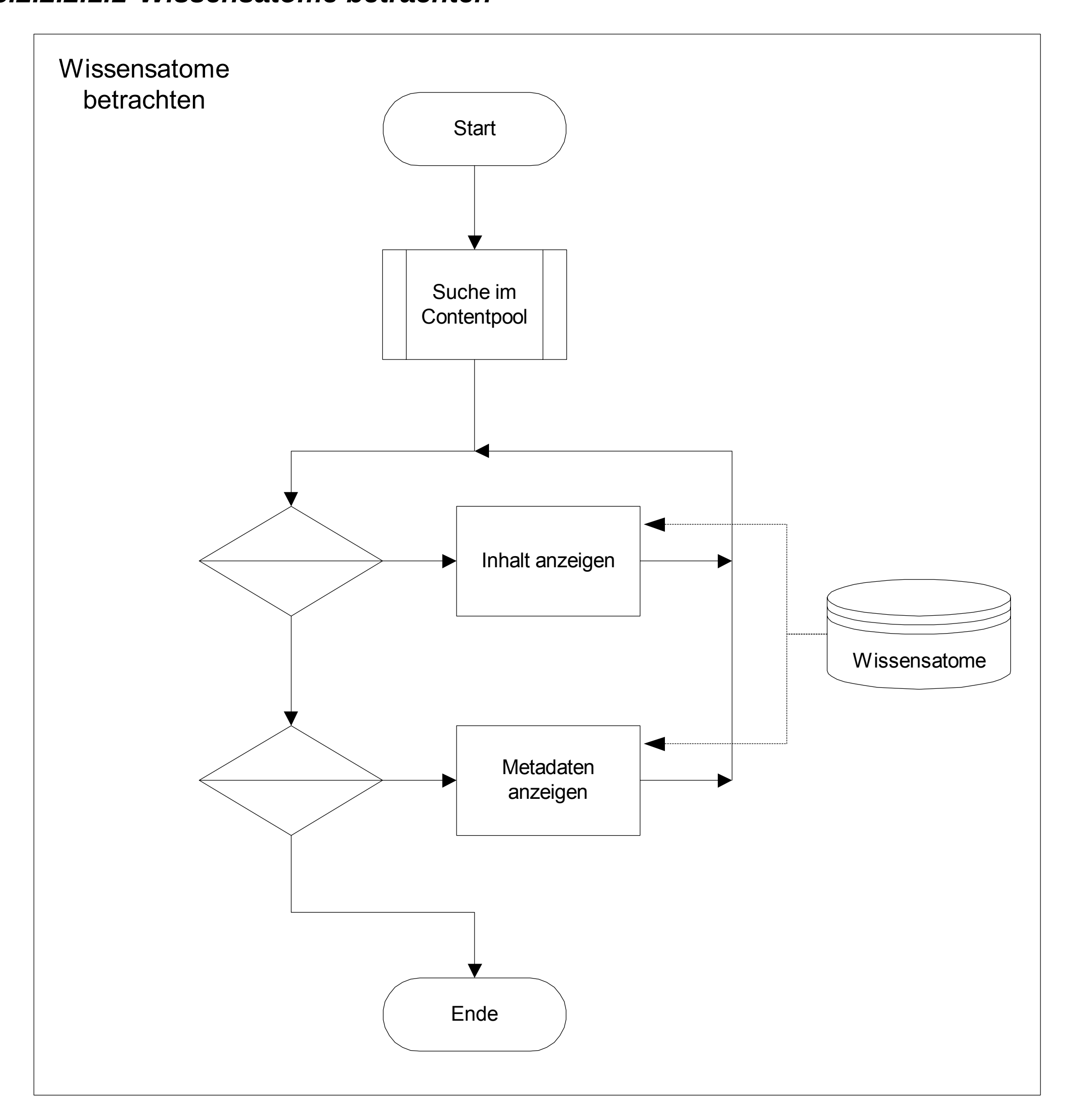


Abbildung 78: Präsentations-Tool: Wissensatome betrachten

6.2.2.2.1.2 Topics betrachten

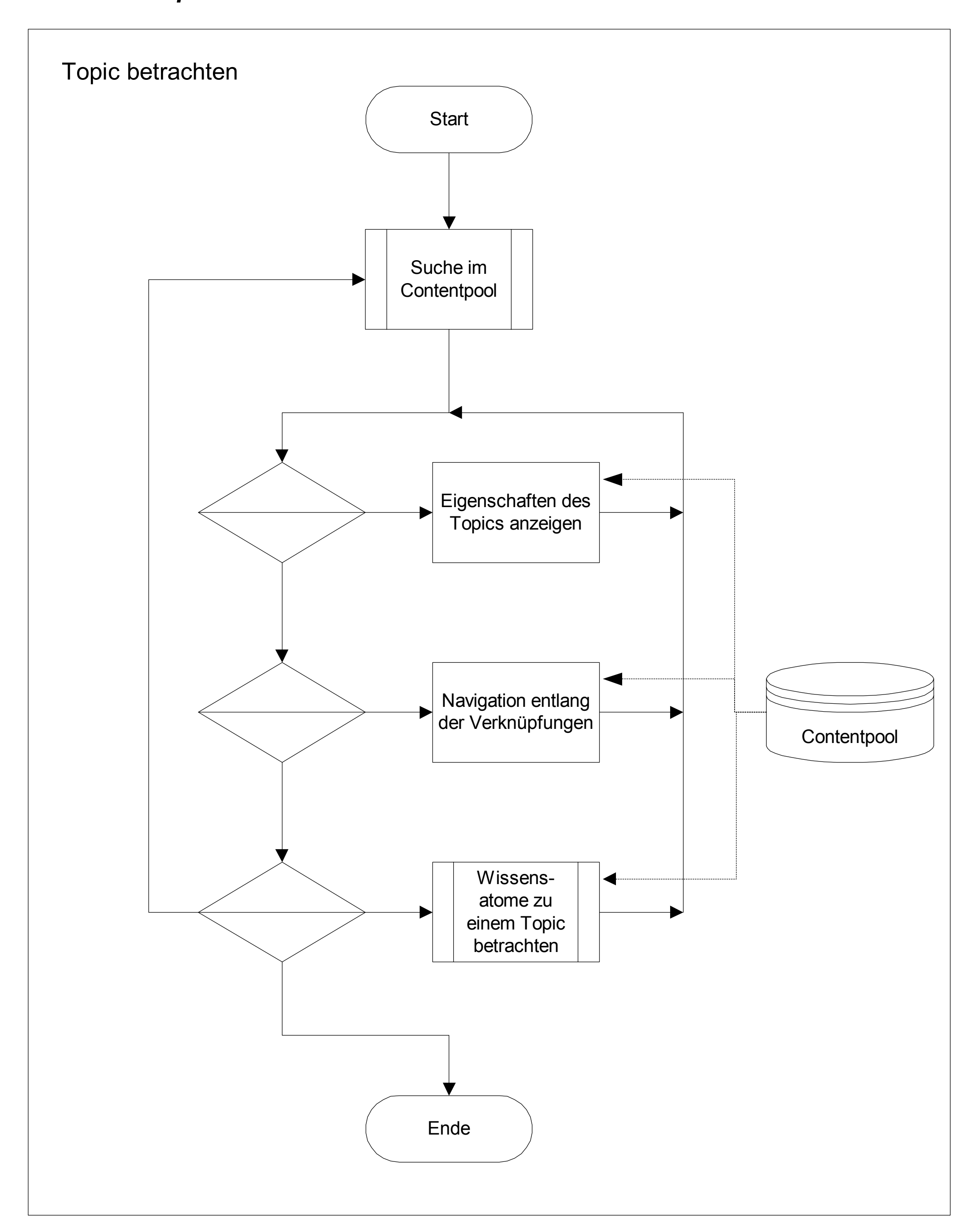


Abbildung 79: Präsentations-Tool: Topics betrachten

6.2.2.2.1.3 Persönliches Nachschlagewerk erstellen

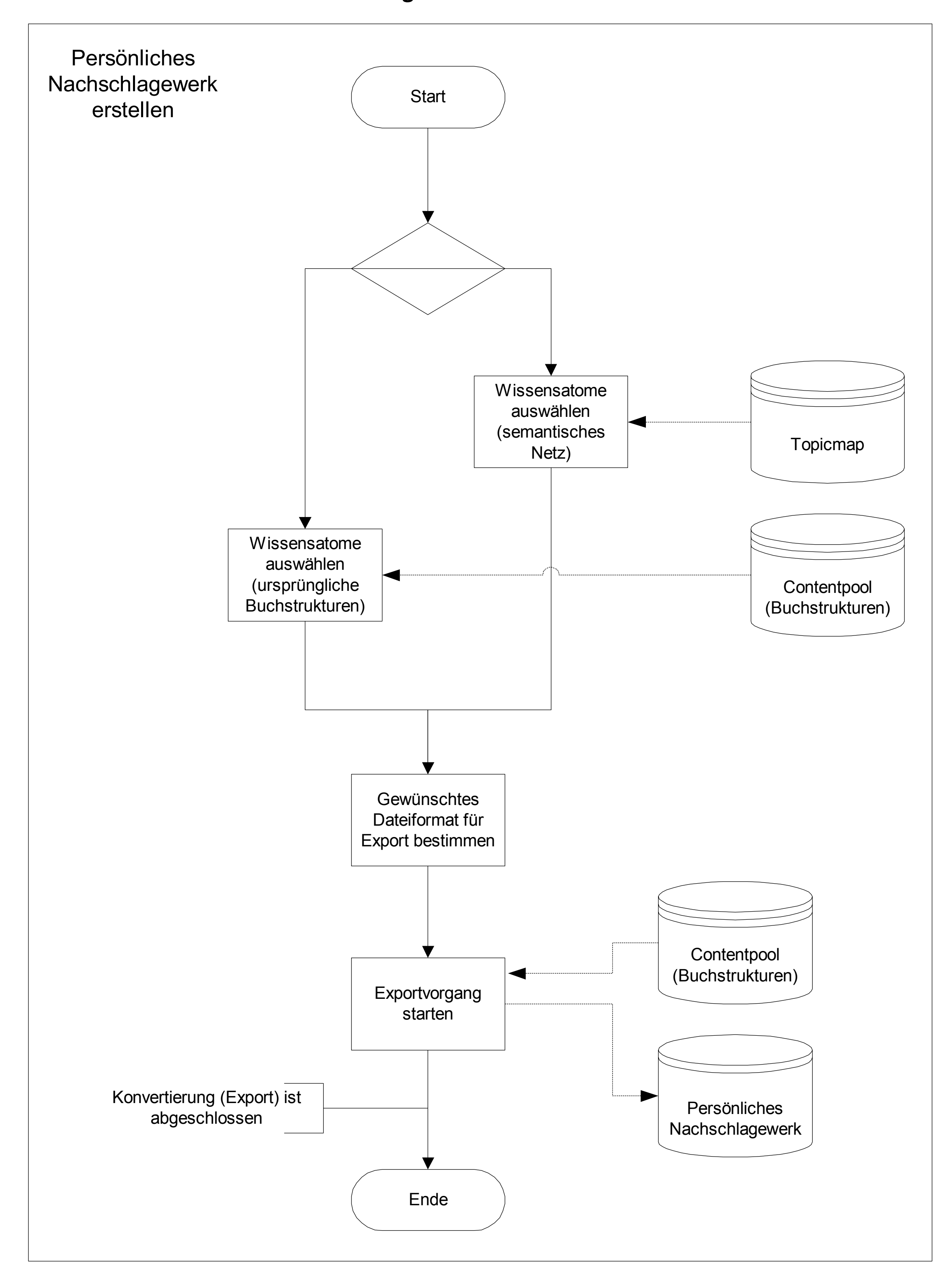


Abbildung 80: Präsentations-Tool: persönliches Nachschlagewerk erstellen

6.2.2.2.2 *Wissensatome verwalten*

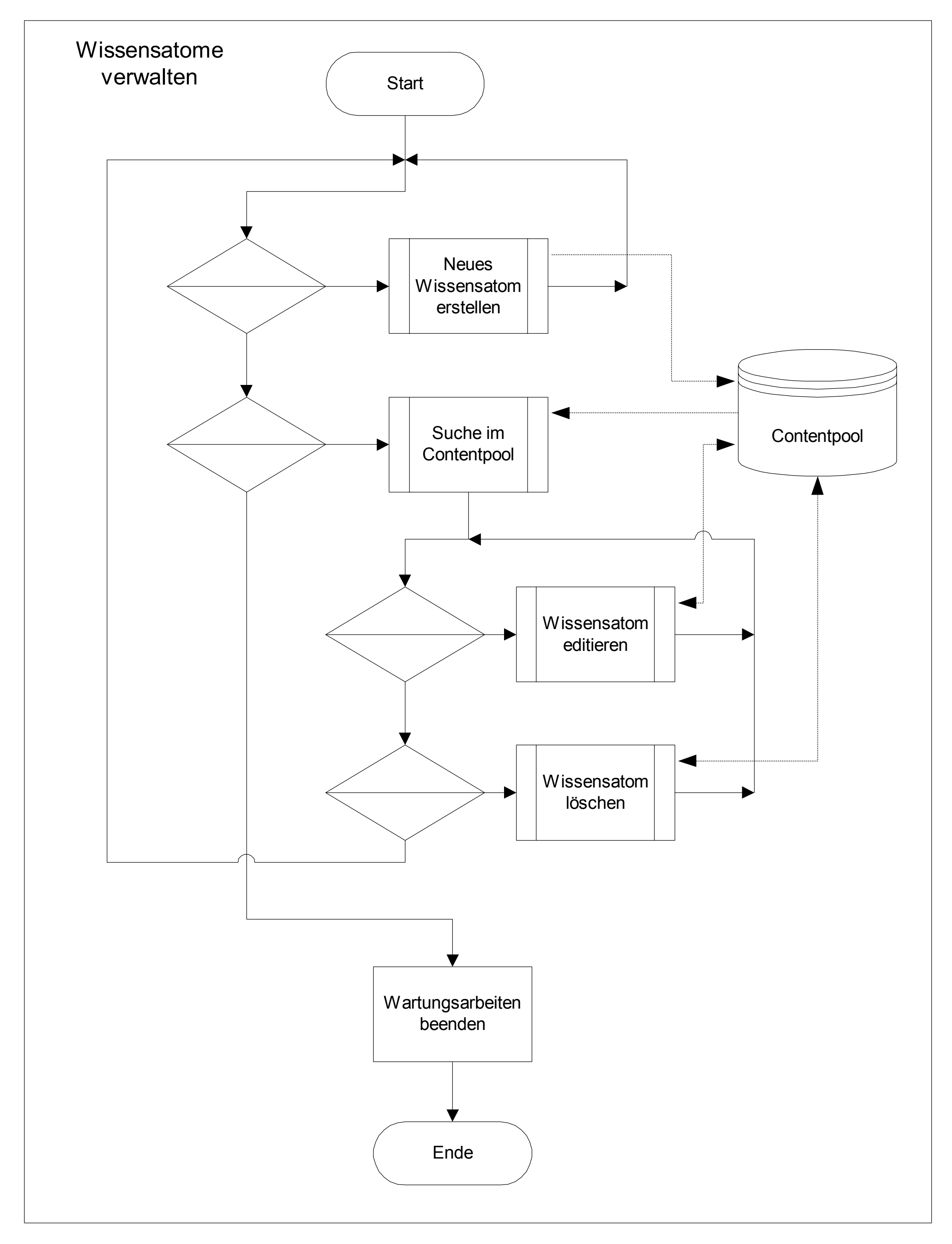


Abbildung 81: Präsentations-Tool: Wissensatome verwalten

6.2.2.2.2.1 Wissensatom erstellen

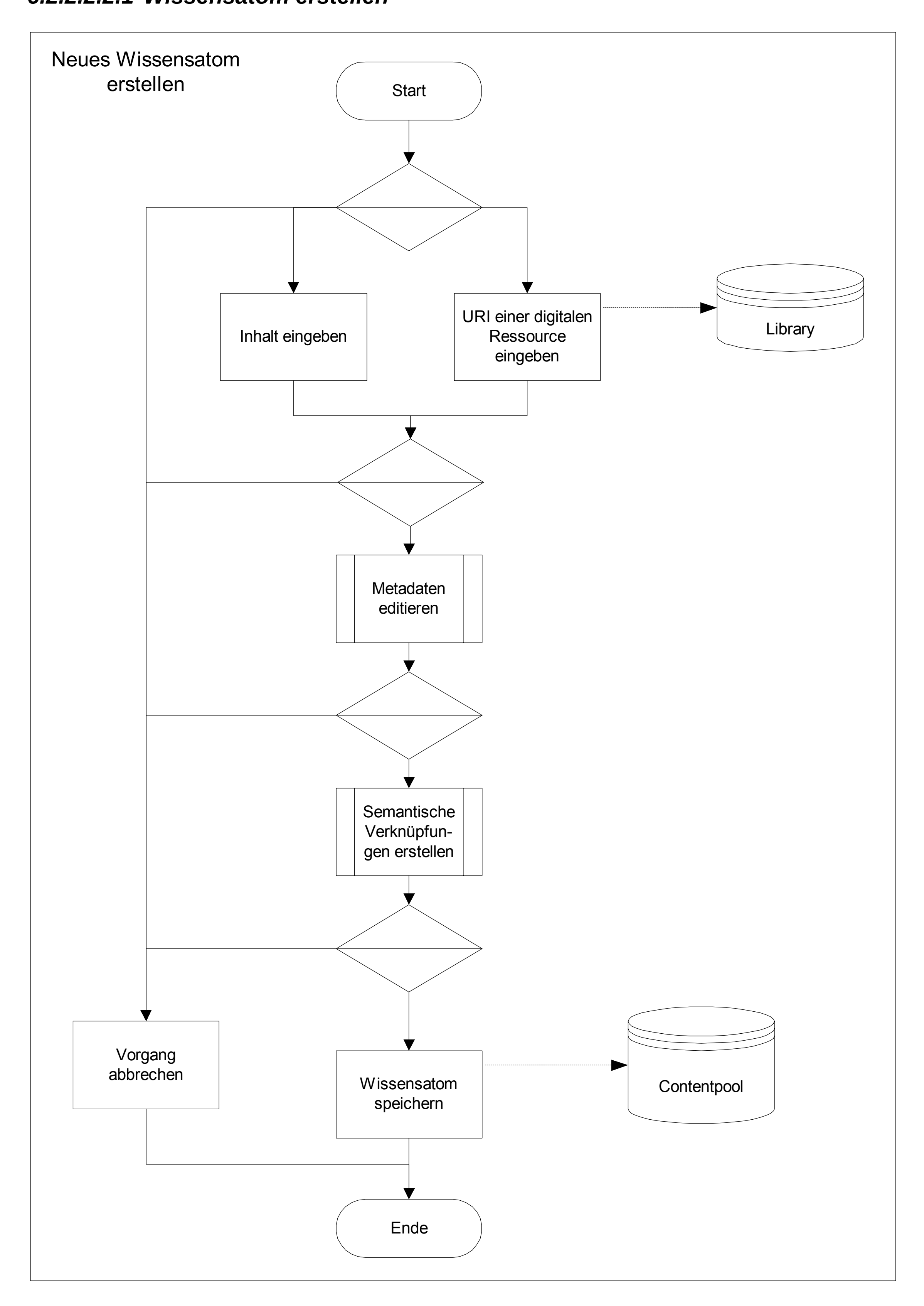


Abbildung 82: Wissensatom-Verwaltung: Wissensatom erstellen

6.2.2.2.2.2 Wissensatom editieren

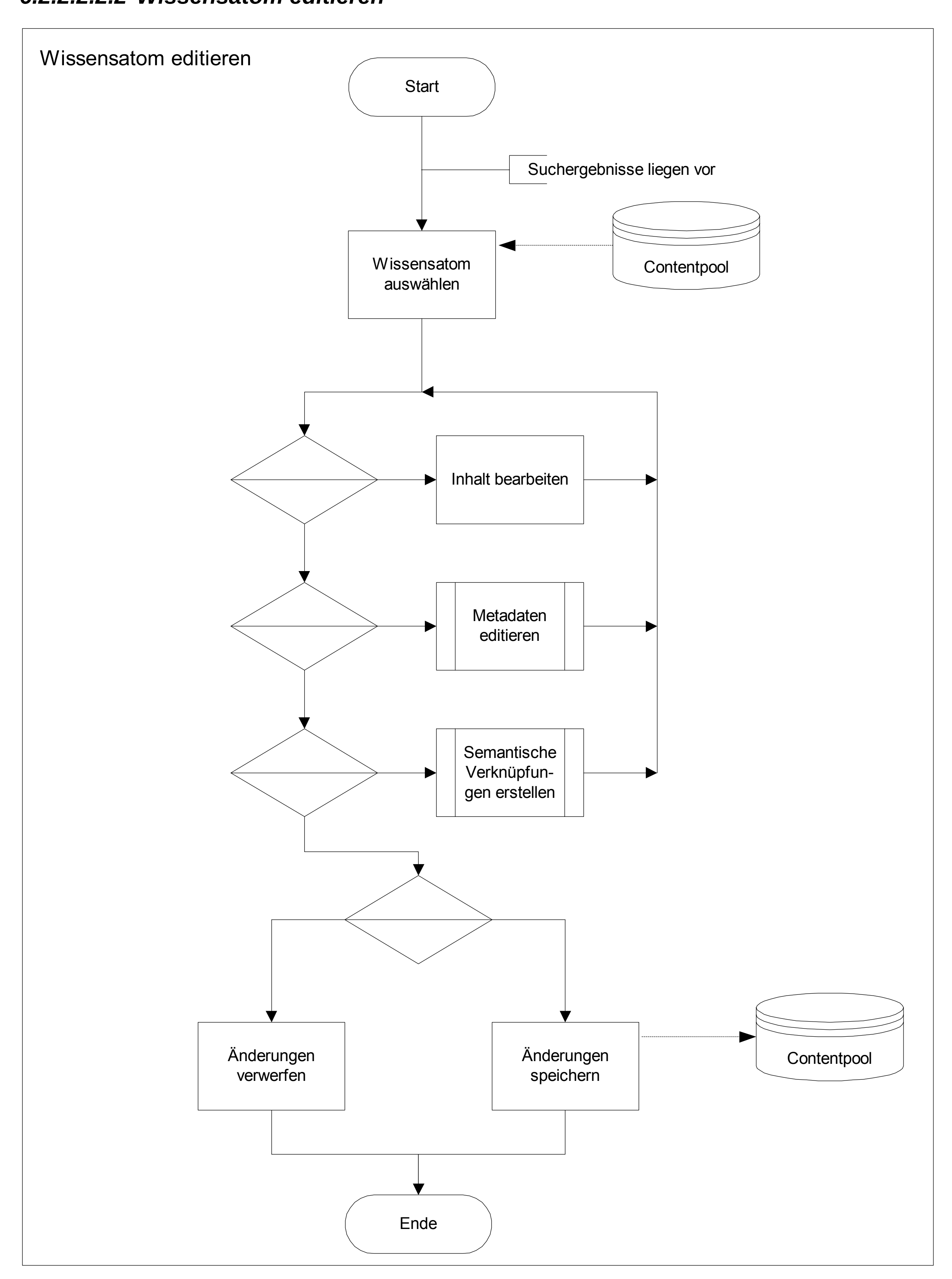


Abbildung 83: Wissensatom-Verwaltung: Wissensatom editieren

6.2.2.2.2.3 Metadaten editieren

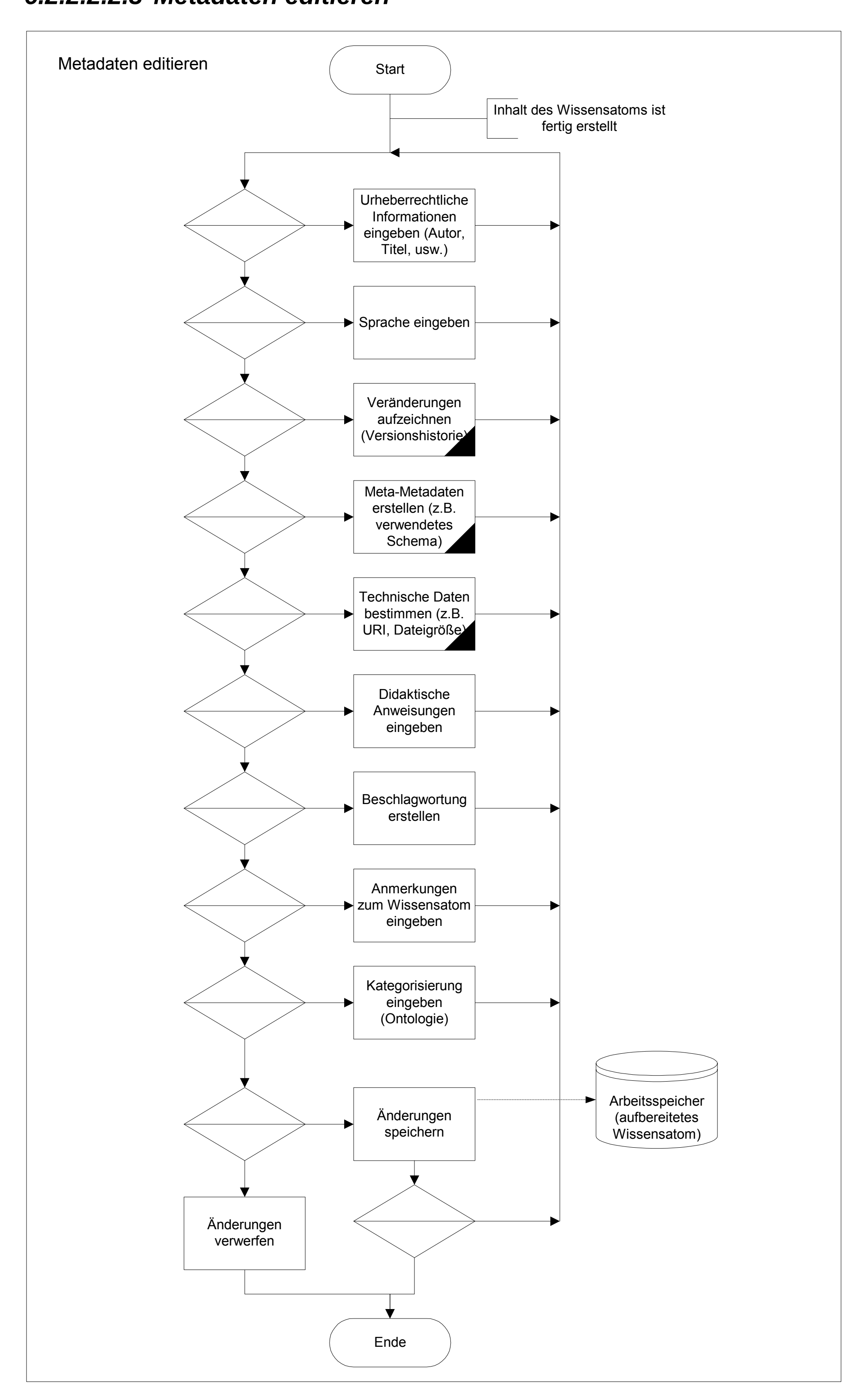


Abbildung 84: Wissensatom editieren (Wissensatom-Verwaltung): Metadaten editieren

6.2.2.2.2.4 Semantische Verknüpfungen erstellen

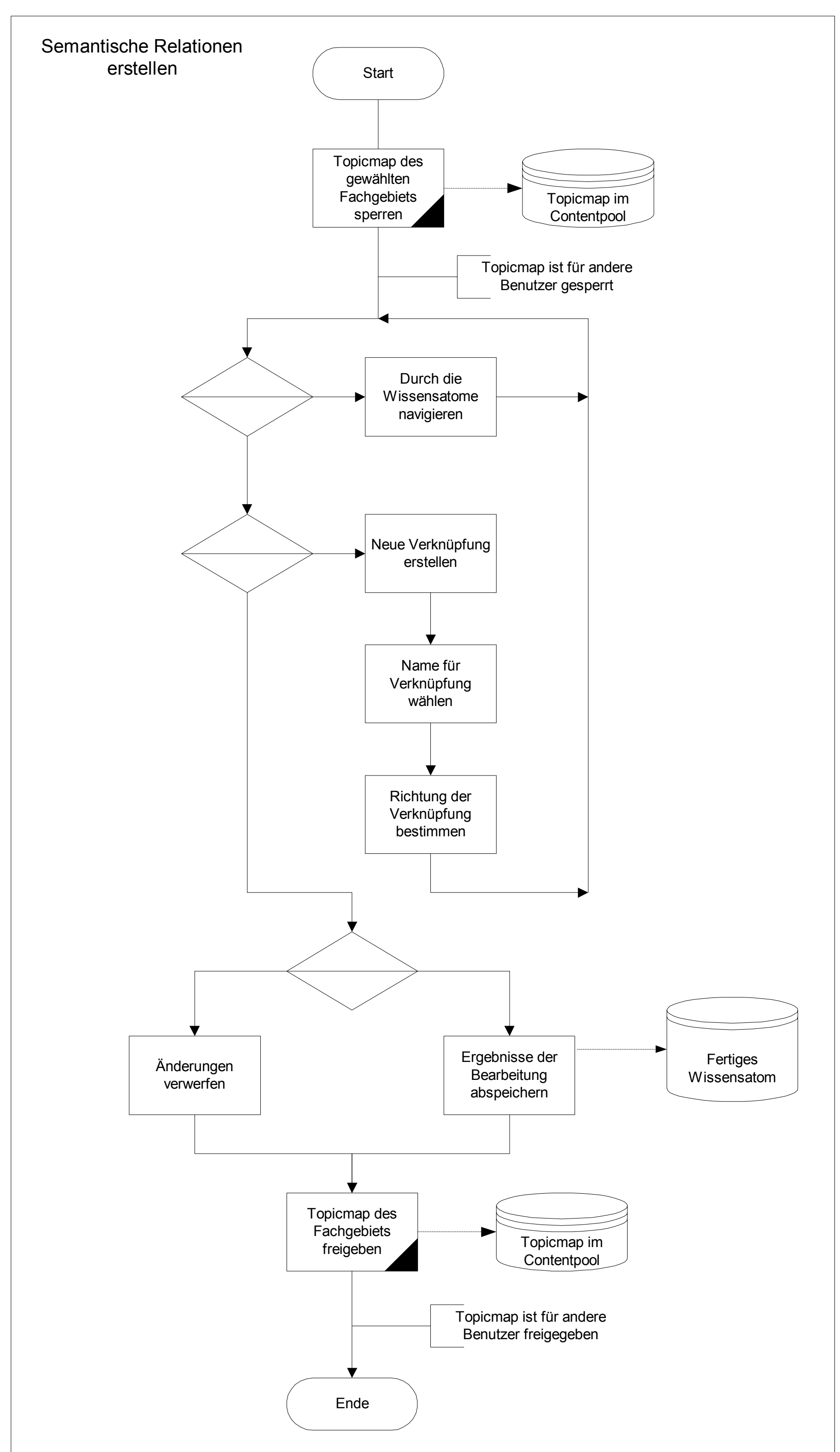


Abbildung 85: Wissensatom editieren (Wissensatom-Verwaltung): Semantische Verknüpfungen erstellen

6.2.2.2.2.5 Wissensatom löschen

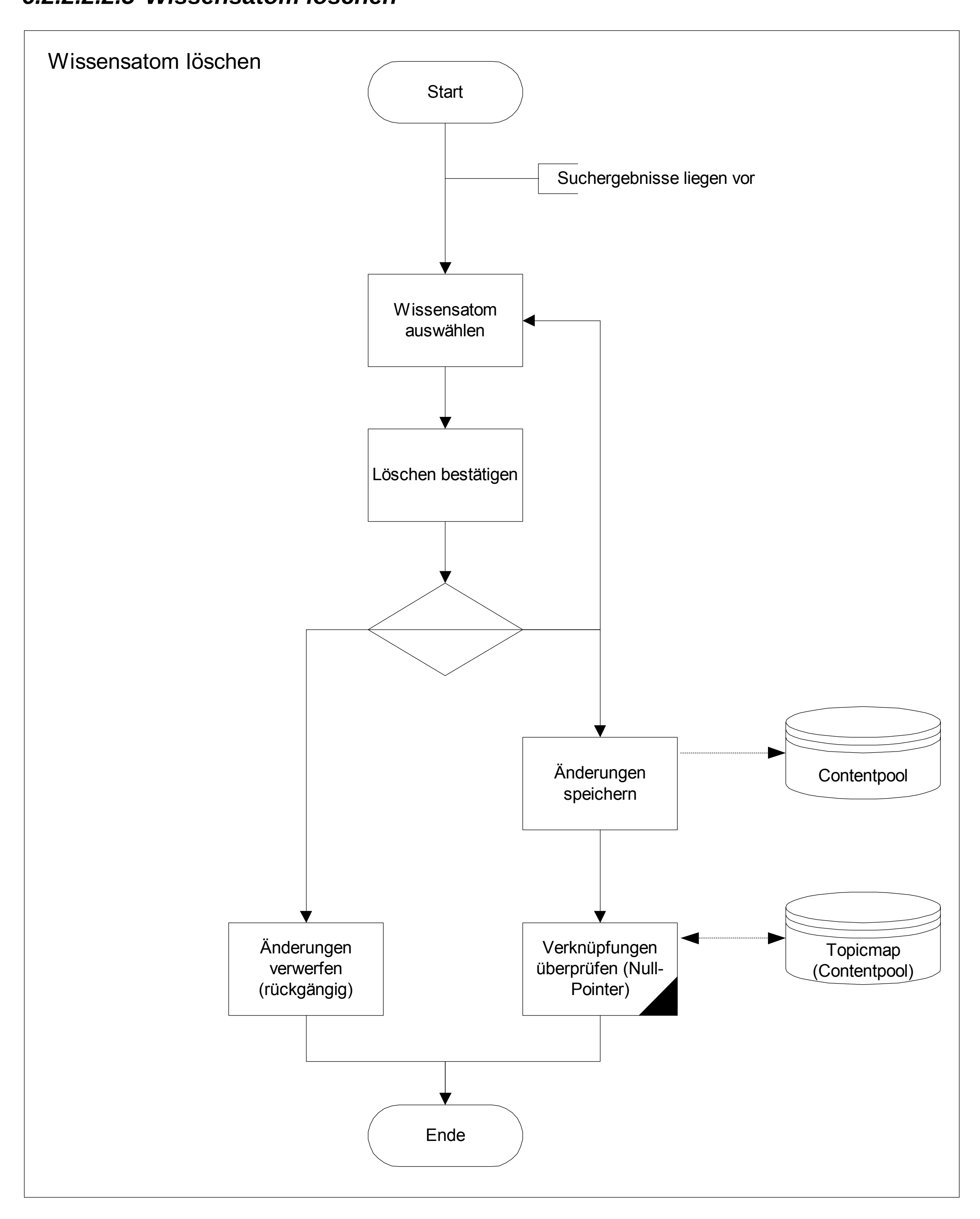


Abbildung 86: Wissensatom-Verwaltung: Wissensatom löschen

6.2.2.2.3 Topics verwalten

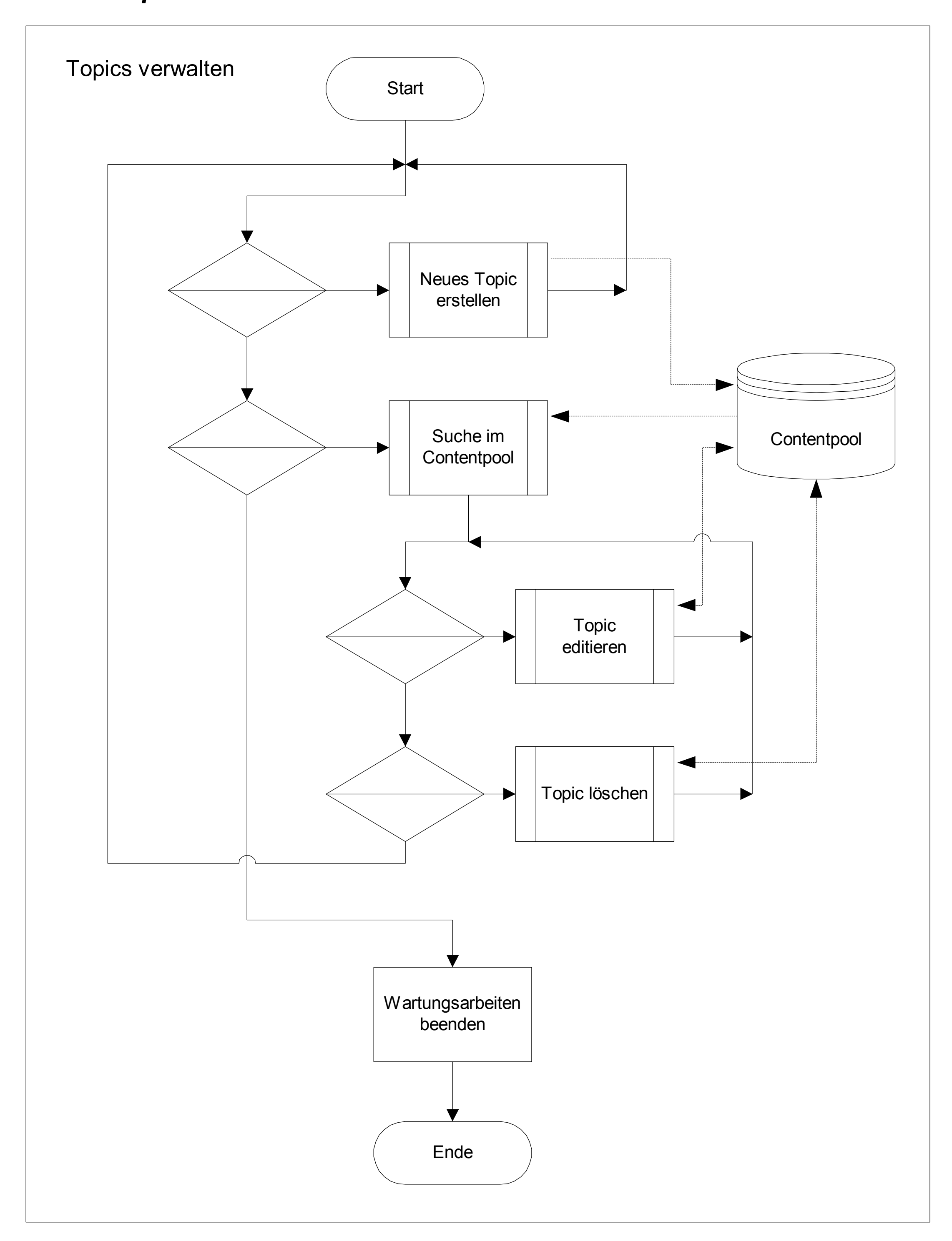


Abbildung 87: Präsentations-Tool: Topics verwalten

6.2.2.2.3.1 Topic erstellen

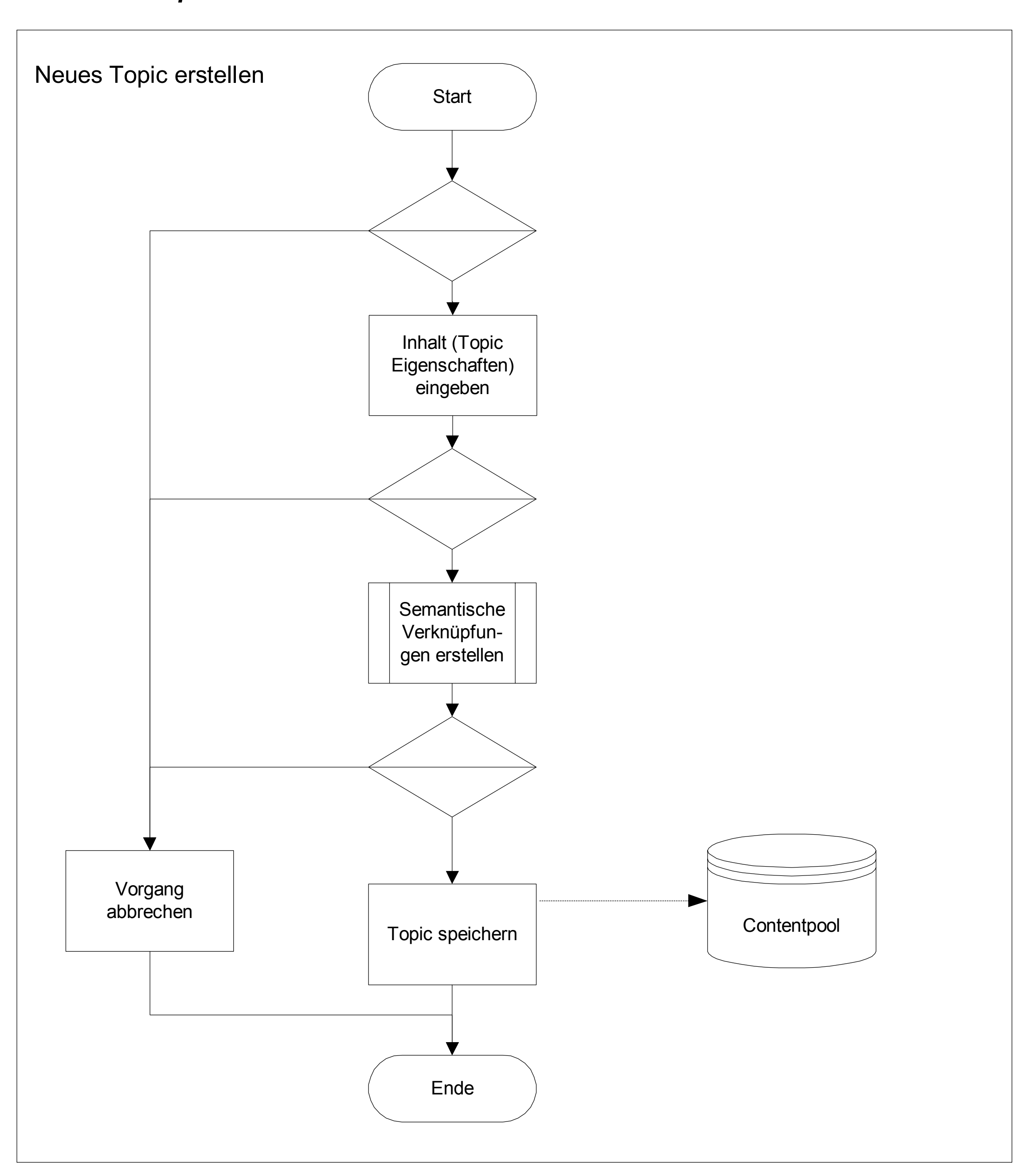


Abbildung 88: Topic-Verwaltung: Topic erstellen

6.2.2.2.3.2 Topic editieren

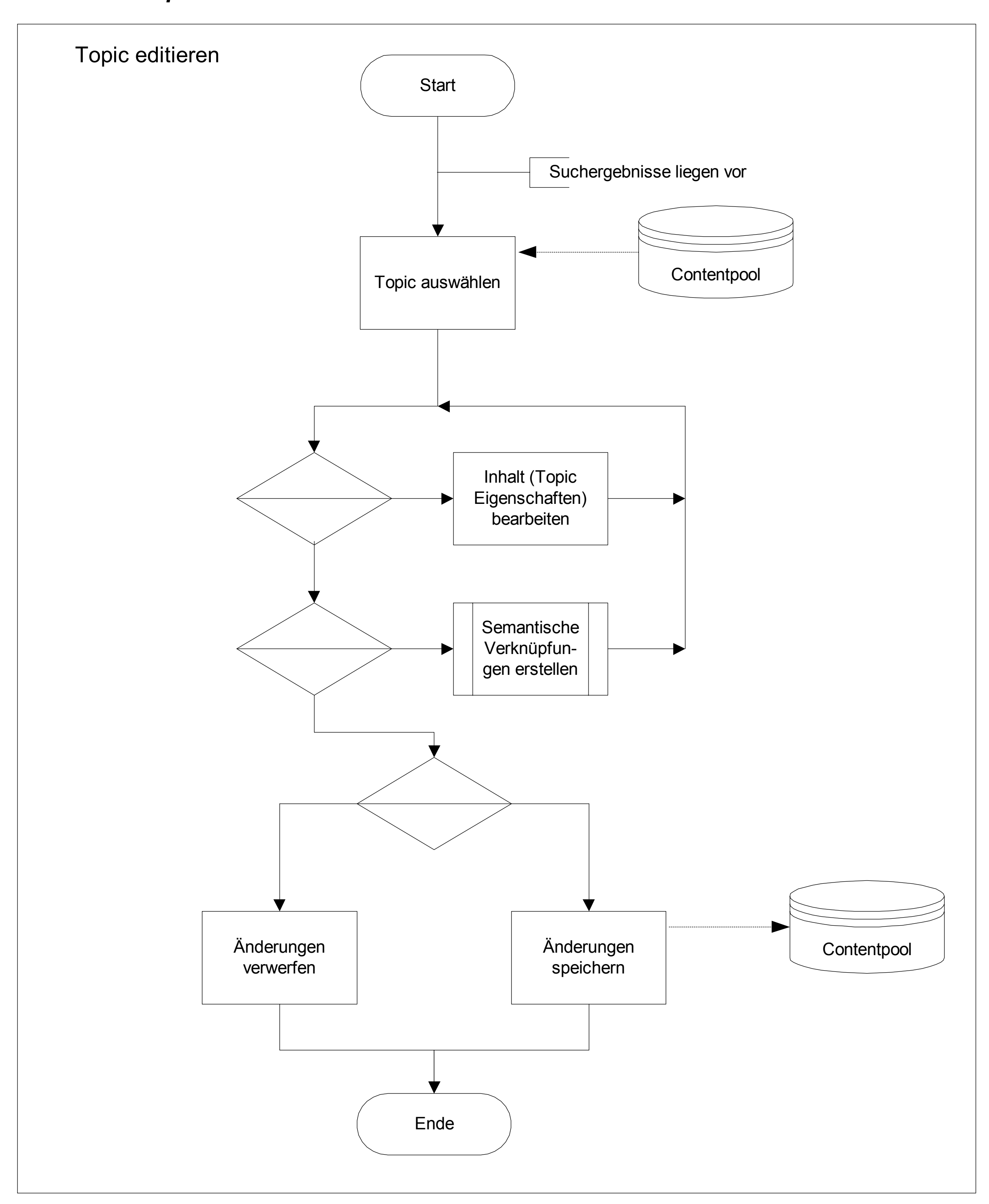


Abbildung 89: Topic-Verwaltung: Topic editieren

6.2.2.2.3.3 Semantische Verknüpfungen erstellen

Semantische Relationen erstellen

Start

Topicmap des gewählten Fachgebiets sperren

Topicmap im Contentpool

Topicmap ist für andere Benutzer gesperrt

Durch die Topics der Topicmap navigieren

Neue Verknüpfung erstellen

Name für Verknüpfung wählen

Richtung der Verknüpfung bestimmen

Änderungen verwerfen

Ergebnisse der Bearbeitung abspeichern

Fertiges Topic

Topicmap des Fachgebiets freigeben

Topicmap im Contentpool

Topicmap ist für andere Benutzer freigegeben

Ende

Abbildung 90: Topic editieren (Topic-Verwaltung): Semantische Verknüpfungen erstellen

6.2.2.2.3.4 Topic löschen

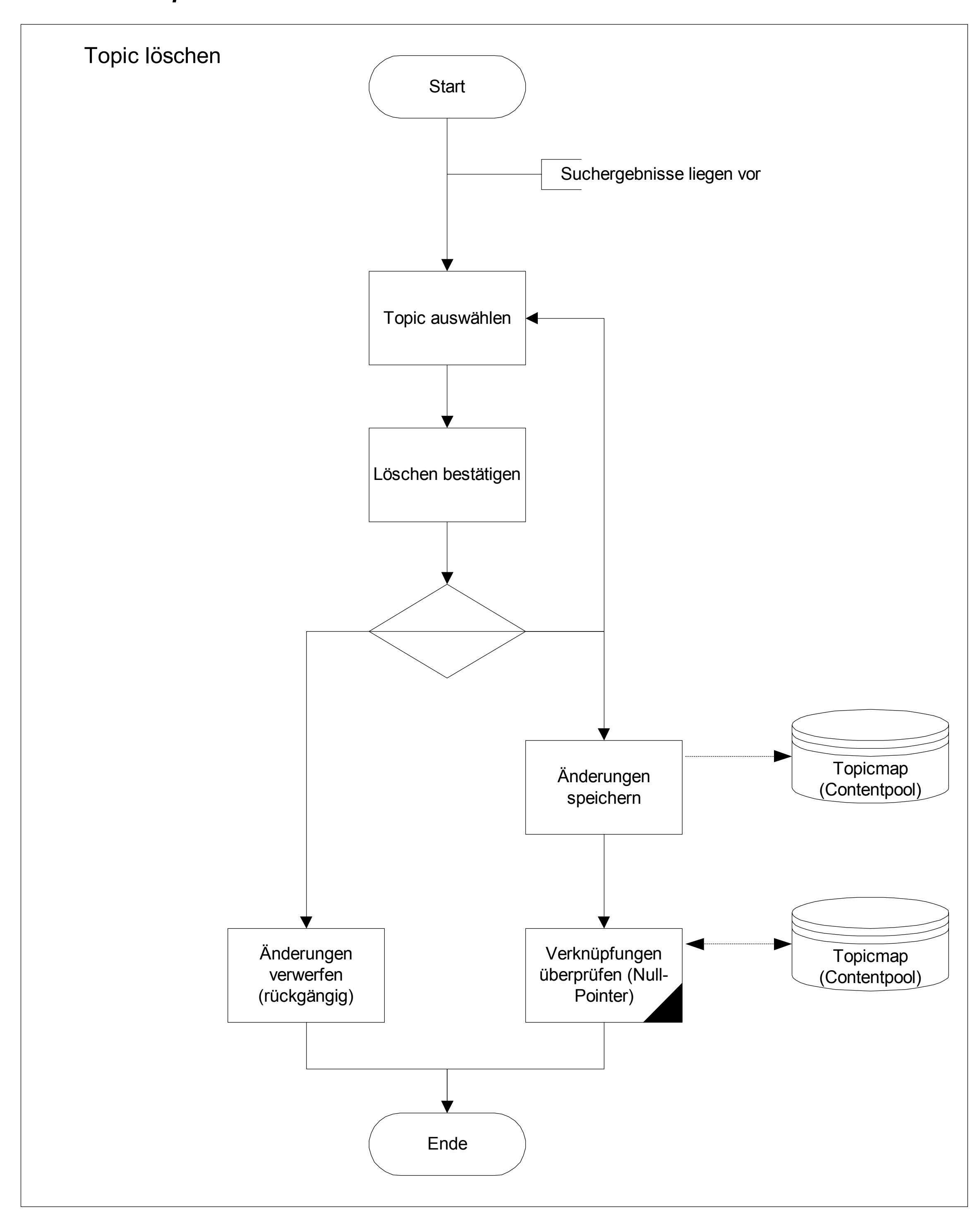


Abbildung 91: Topic-Verwaltung: Topic löschen

6.2.2.3 Topicmap-Manager

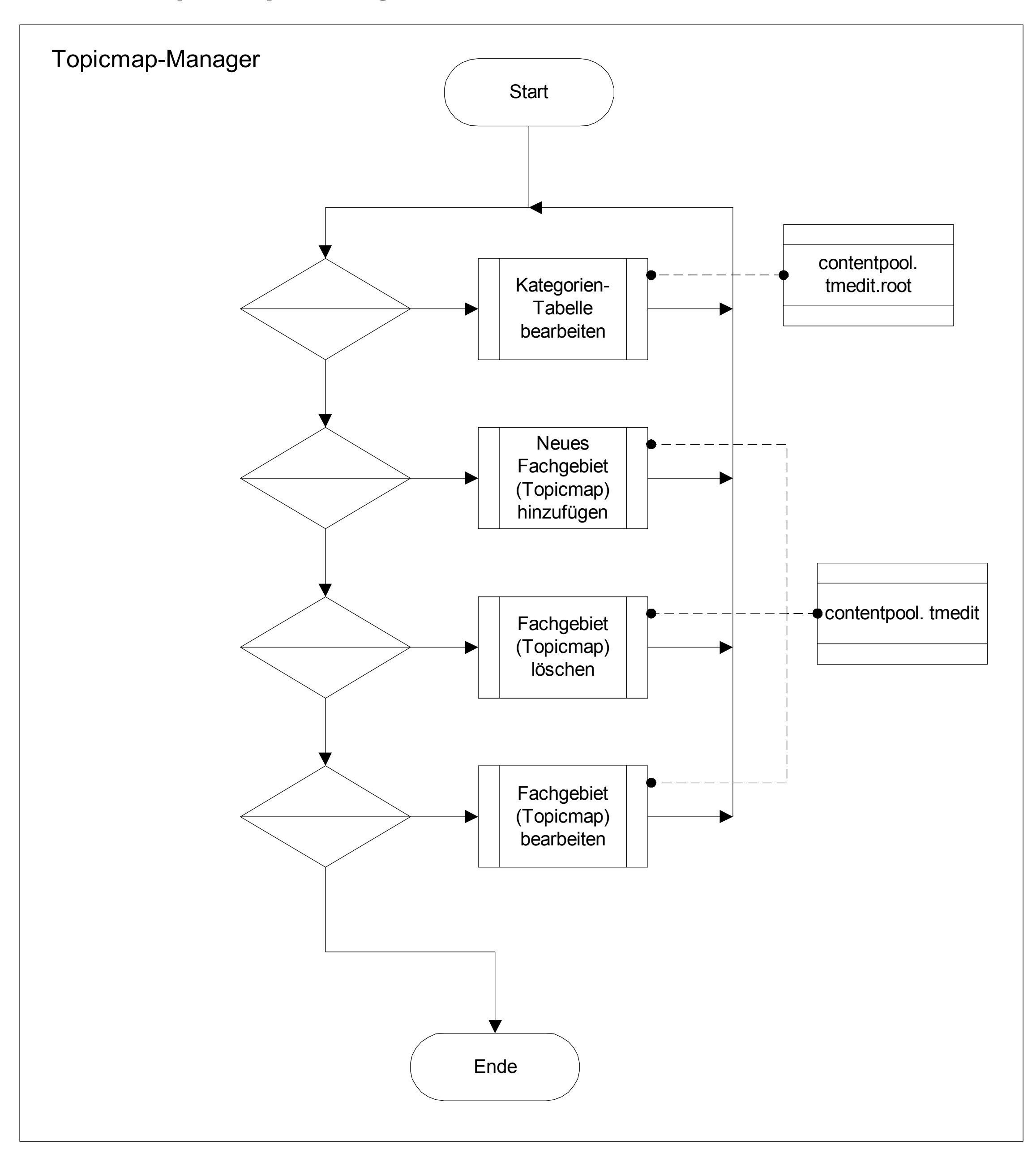


Abbildung 92: Modell des Topicmap-Managers

Beschreibung: Ontologien sind derjenige Teil des Contentpools, der das **Bindeglied zwischen verschiedenen Fachgebieten** darstellt (vgl. die Definition des Begriffs im Kapitel 10.2 Glossar, Seite 333 ff.). D.h. in der **allgemeinen Ontologie** des Contentpools sind die Konzepte abgebildet, die für alle im Contentpool vorhandenen Fachgebiete und damit auch für die semantischen Netze gelten. (Vgl. dazu den Abschnitt 6.1.2 Konzeptuelle Architektur des Contentpools, Seite 147 ff.) Der Topicmap-Manager dient zur Verwaltung sowohl der **allgemeinen Ontologie** (d.h. der so genannten Kategorien-Tabelle) als auch der **speziellen Ontologien**,

die für jedes Fachgebiet durch einen Administrator des Contentpools zu definieren sind. Folgende Features werden durch den Topicmap-Manager unterstützt: Kategorien-Tabelle bearbeiten, Ontologie eines Fachgebiets hinzufügen, Ontologie eines Fachgebiets löschen oder bearbeiten. Dabei werden Ontologien eines Fachgebiets durch Topic Maps abgebildet (vgl. zur Theorie über Topic Maps das Kapitel 4.4.2: XML Topic Maps (XTM), Seite 77 ff. sowie zur Realisierung im Contentpool den Abschnitt 6.1.2: Konzeptuelle Architektur des Contentpools, Seite 147 ff., besonders Abbildung 50: Konzeptuelles Modell des allgemeinen Contentpools und Abbildung 51: Verknüpfung semantischer Netze mit Hilfe von Ontologien).

Kategorien-Tabelle bearbeiten: Sämtliche Konzepte, die allen Fachgebieten im Contentpool gemeinsam sind, werden in der Kategorien-Tabelle gespeichert. Die Kategorien-Tabelle ist eine besondere Topic Map, in welcher die allgemeine Ontologie definiert wird. Unter der allgemeinen Ontologie wird eine **Sammlung von Begriffen zu Konzepten** verstanden, die auf dem Gebiet unterschiedlicher Wissenschaften von Bedeutung sind oder sein können. (Vgl. dazu auch die Definition des Begriffs „Ontologie" im Glossar, Seite 340.) In der Kategorien-Tabelle wird für jeden allgemeinen Begriff zu einem Konzept ein Topic erstellt.

Um diese Tabelle auf dem neuesten Stand halten zu können, wird Funktionalität (1) zum **Hinzufügen** eines allgemeinen Begriffs, (2) zum **Entfernen** eines allgemeinen Begriffs und (3) zum **Bearbeiten** eines bestehenden allgemeinen Begriffs geboten. Vorgenommene Änderungen werden in die Kategorien-Tabelle geschrieben und wirken sich unter Umständen auch auf die semantischen Netze im Contentpool aus. Änderungen sollten daher sorgfältig vorgenommen werden – unsachgemäße Manipulationen können die Brauchbarkeit des Contentpools negativ beeinflussen!

Neues Fachgebiet hinzufügen: ein neues Fachgebiet hinzufügen bedeutet, dass zuerst der **Name** und die **Beschreibung** des Fachgebiets eingegeben werden muss. Anschließend ist eine **spezielle Ontologie** für das Fachgebiet zu definieren – d.h. die Konzepte der allgemeinen Ontologie sind für das neue Fachgebiet spezifisch zu benennen und zu beschreiben. Konzepte einer speziellen Ontologie werden in einer Topic Map modelliert, welche mit der allgemeinen Ontologie verknüpft ist. Dabei besteht das Ziel darin, für möglichst alle der in der allgemeinen Ontologie definierten allgemeinen Begriffe zu Konzepten eine **Ausprägung** zu definieren, die für das aktuelle Fachgebiet **spezifisch und gültig** ist. Nach Abschluss dieser Schritte steht ein neues, noch leeres semantisches Netz zur Verfügung, welches mit Wissensatomen „gefüllt" werden kann.

Bestehendes Fachgebiet löschen: dem Benutzer wird eine Liste der im Contentpool vorhandenen Fachgebiete angezeigt. Nach **Auswahl** eines Fachgebiets kann der Befehl gegeben werden, dieses zu löschen. Dadurch wird vor-

erst das semantische Netz zu diesem Fachgebiet entfernt. **Achtung**: im semantischen Netz ist der Großteil des Mehrwerts gespeichert, den der Contentpool gegenüber herkömmlichen Lehrbüchern besitzt. Mit dem Löschbefehl muss daher vorsichtig umgegangen werden! Wird ein semantisches Netz tatsächlich gelöscht, so besteht anschließend die Möglichkeit, die damit verknüpften **Wissensatome beizubehalten** oder ebenfalls zu **löschen**. Alle Befehle können wieder **rückgängig gemacht** werden. Diese Möglichkeit besteht nicht mehr, sobald die Änderungen gespeichert wurden. Dies muss mit einem eigenen Befehl geschehen.

Bestehendes Fachgebiet bearbeiten: Beim Bearbeiten eines Fachgebiets besteht die Möglichkeit, den **Namen** des Fachgebiets, seine **Beschreibung** oder die Ontologie, d.h. die **spezifischen Bezeichnungen** (Ausprägungen) der allgemeinen Begriffe (für Konzepte) zu verändern.

In der Folge werden die Prozessmodelle für die genannten und beschriebenen Tools und Features präsentiert:

- 6.2.2.3.1 Kategorien-Tabelle bearbeiten .... Seite 209
- 6.2.2.3.2 Fachgebiet (Topic Map) hinzufügen .... Seite 210
- 6.2.2.3.3 Fachgebiet (Topic Map) löschen .... Seite 211
- 6.2.2.3.4 Fachgebiet (Topic Map) bearbeiten .... Seite 212

6.2.2.3.1 Kategorien-Tabelle bearbeiten

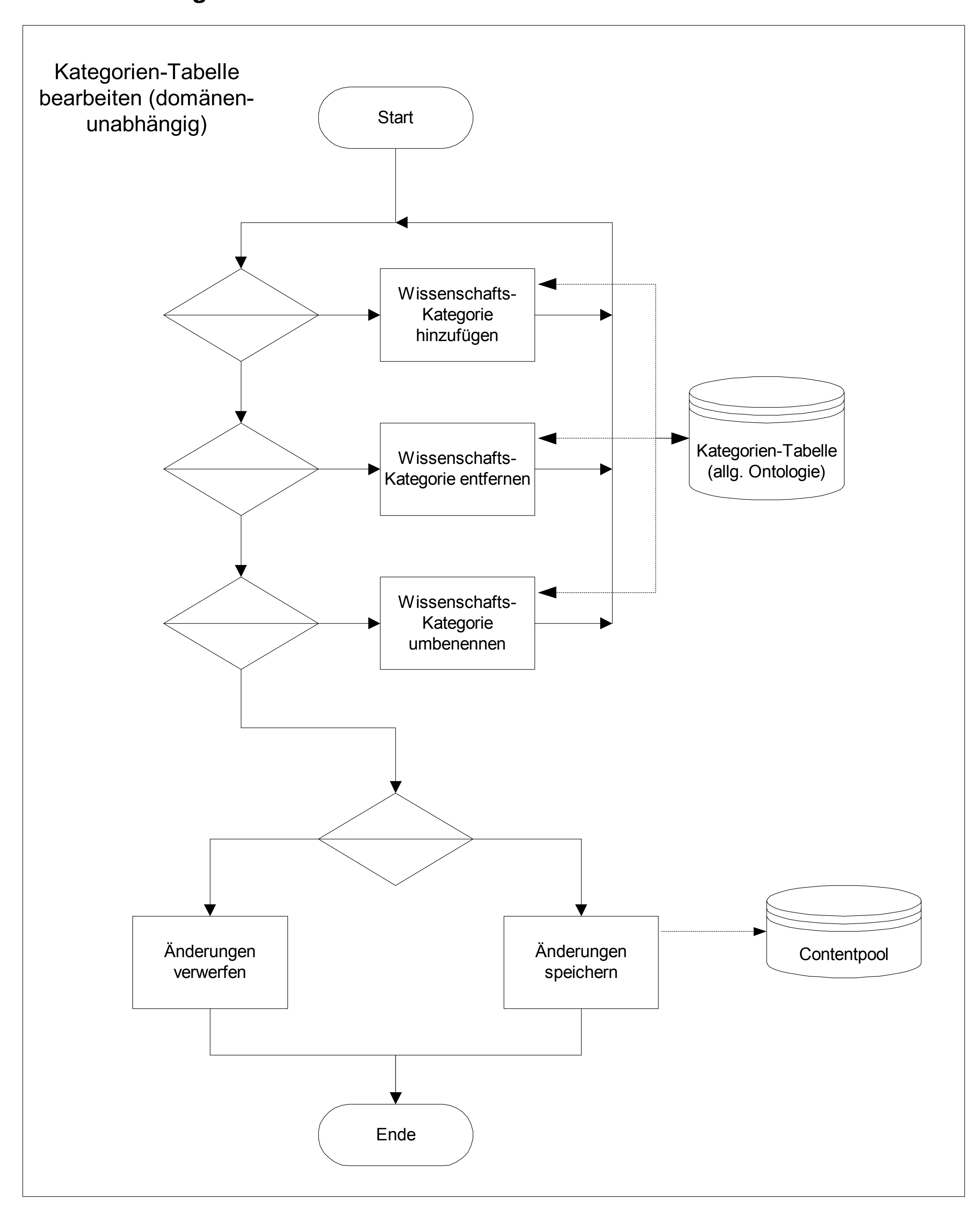


Abbildung 93: Topicmap-Manager: Kategorien-Tabelle bearbeiten

6.2.2.3.2 Fachgebiet (Topic Map) hinzufügen

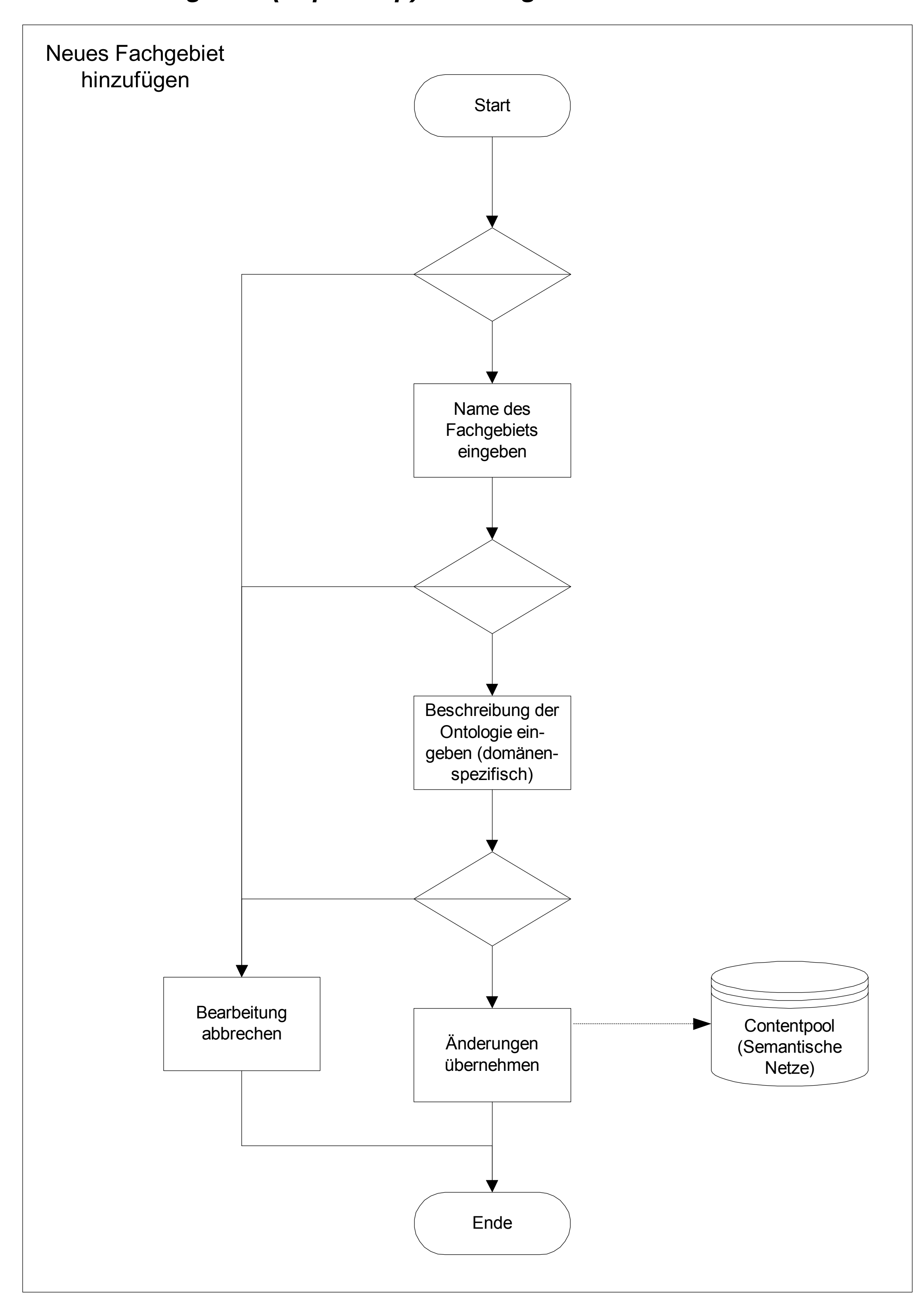


Abbildung 94: Topicmap-Manager: Fachgebiet (Topic Map) hinzufügen

6.2.2.3.3 Fachgebiet (Topic Map) löschen

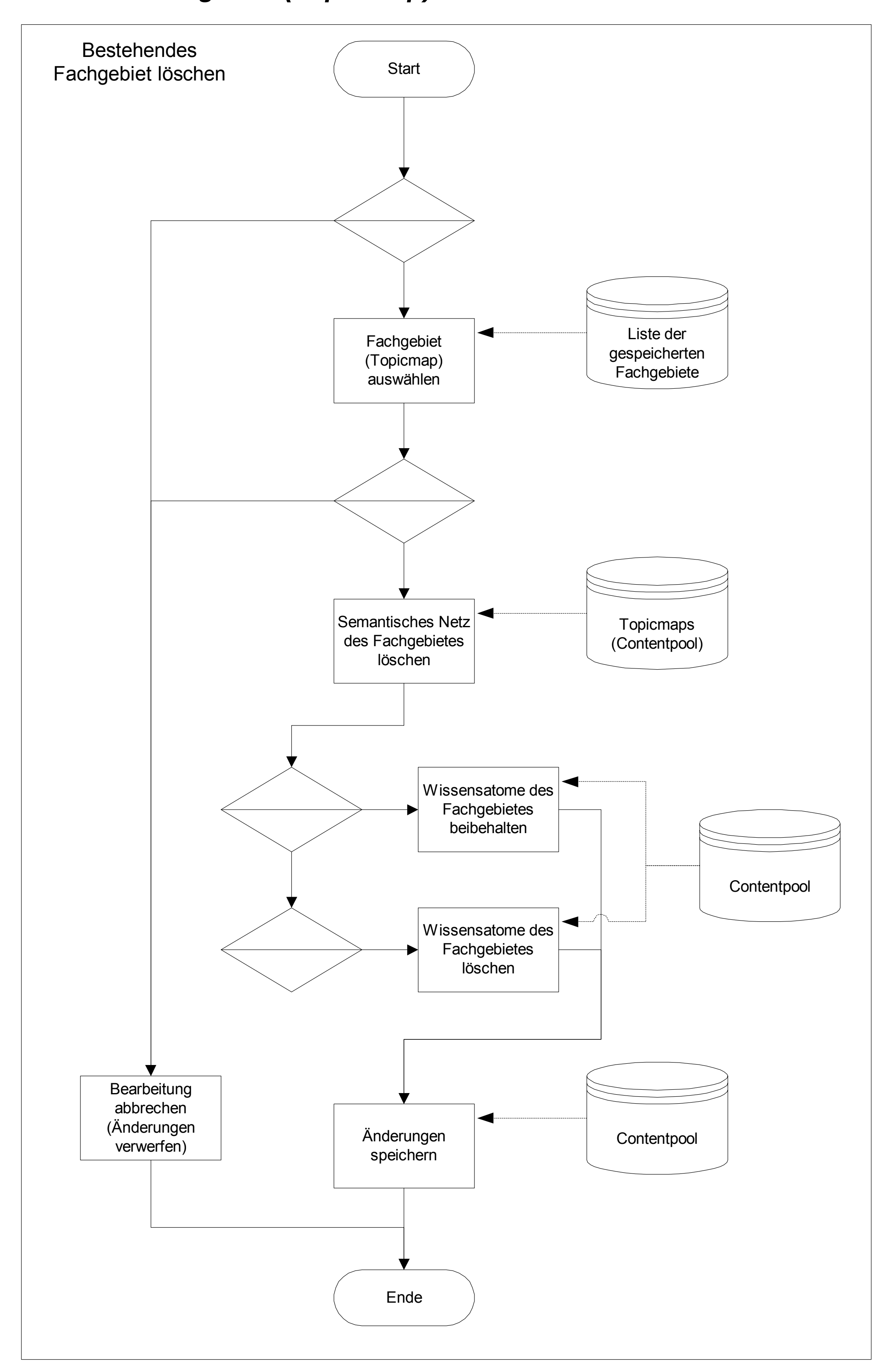


Abbildung 95: Topicmap-Manager: Fachgebiet (Topic Map) löschen

6.2.2.3.4 Fachgebiet (Topic Map) bearbeiten

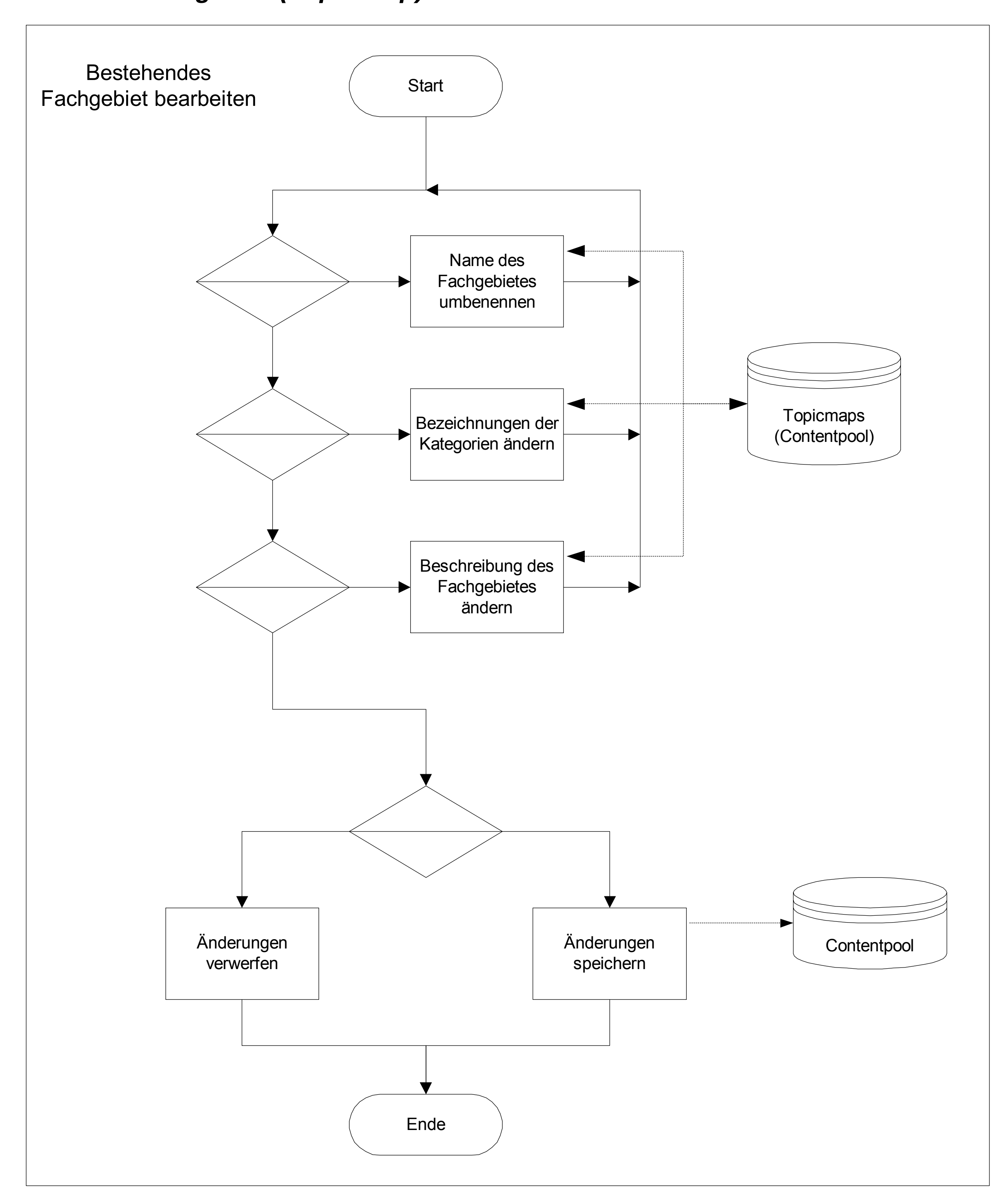


Abbildung 96: Topicmap-Manager: Fachgebiet (Topic Map) bearbeiten

6.2.2.4 Suche im Contentpool

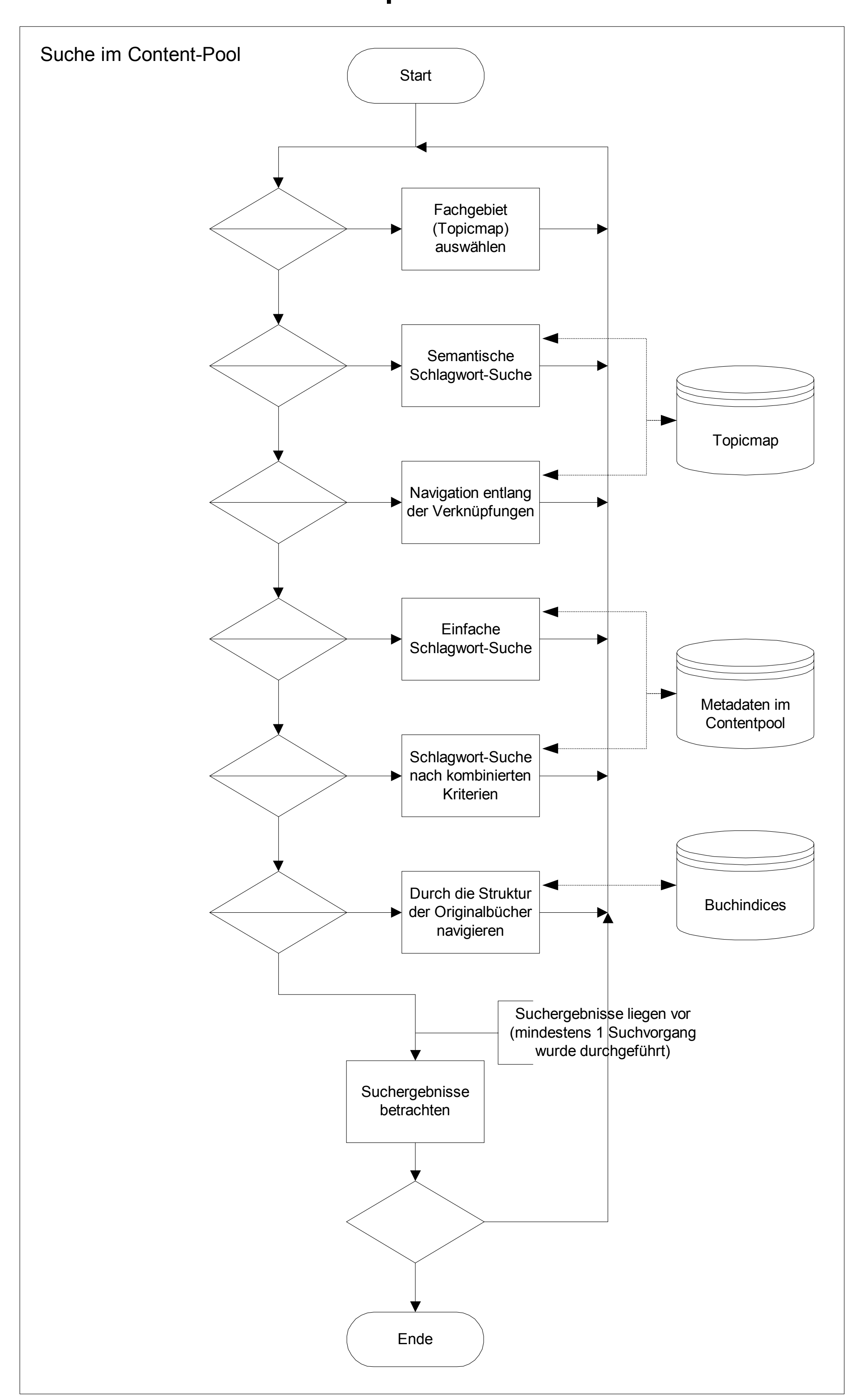


Abbildung 97: Präsentations-Tool: Suche im Contentpool

Beschreibung: die Suche im Contentpool stellt eine der wichtigsten Funktionen dar. Nicht nur als Schnittstelle zum Client-Frontend, sondern auch für die **effiziente Nutzung** der gespeicherten Wissensatome ist die Suchfunktion von enormer Bedeutung. Je umfangreicher die gebotenen Suchmöglichkeiten sind, desto besser wird jeder Benutzer auch die von ihm gewünschten Resultate erzielen und die für ihn geeigneten Wissensatome finden können. Daher bietet das Suchtool die Möglichkeit, in sämtlichen Feldern der **Metadaten** zu suchen, sofern es sich nicht um Information für die interne Bearbeitung des Wissensatoms handelt. Der erste Schritt bei einer Suchabfrage besteht in der **Auswahl eines Fachgebiets**. Alle im Contentpool gespeicherten Fachgebiete (in Form von Topic Maps) stehen dabei zur Auswahl. Anschließend bietet sich dem Benutzer die Möglichkeit, entweder die **ursprünglichen Strukturen** der Lehrbücher oder das **semantische Netz** zu nutzen, oder (ähnlich wie bei herkömmlichen Schlagwortsuchen in elektronischen Dokumenten) innerhalb der Metadaten eine **Schlagwortsuche** durchzuführen. Auch im semantischen Netz kann nach Schlagwörtern gesucht werden, wobei in diesem Fall weniger die syntaktische denn die semantische Dimension der Inhalte berücksichtigt wird. D.h. bei der semantischen Suche werden in erster Linie die Verknüpfungen, bei der herkömmlichen Suche vor allem die Inhalte durchsucht. Bei der herkömmlichen Schlagwortsuche besteht zudem die Möglichkeit, **mehrere Suchbedingungen** mit Hilfe von boole'schen Operatoren beliebig zu kombinieren.

6.2.2.5 Hilfe-System

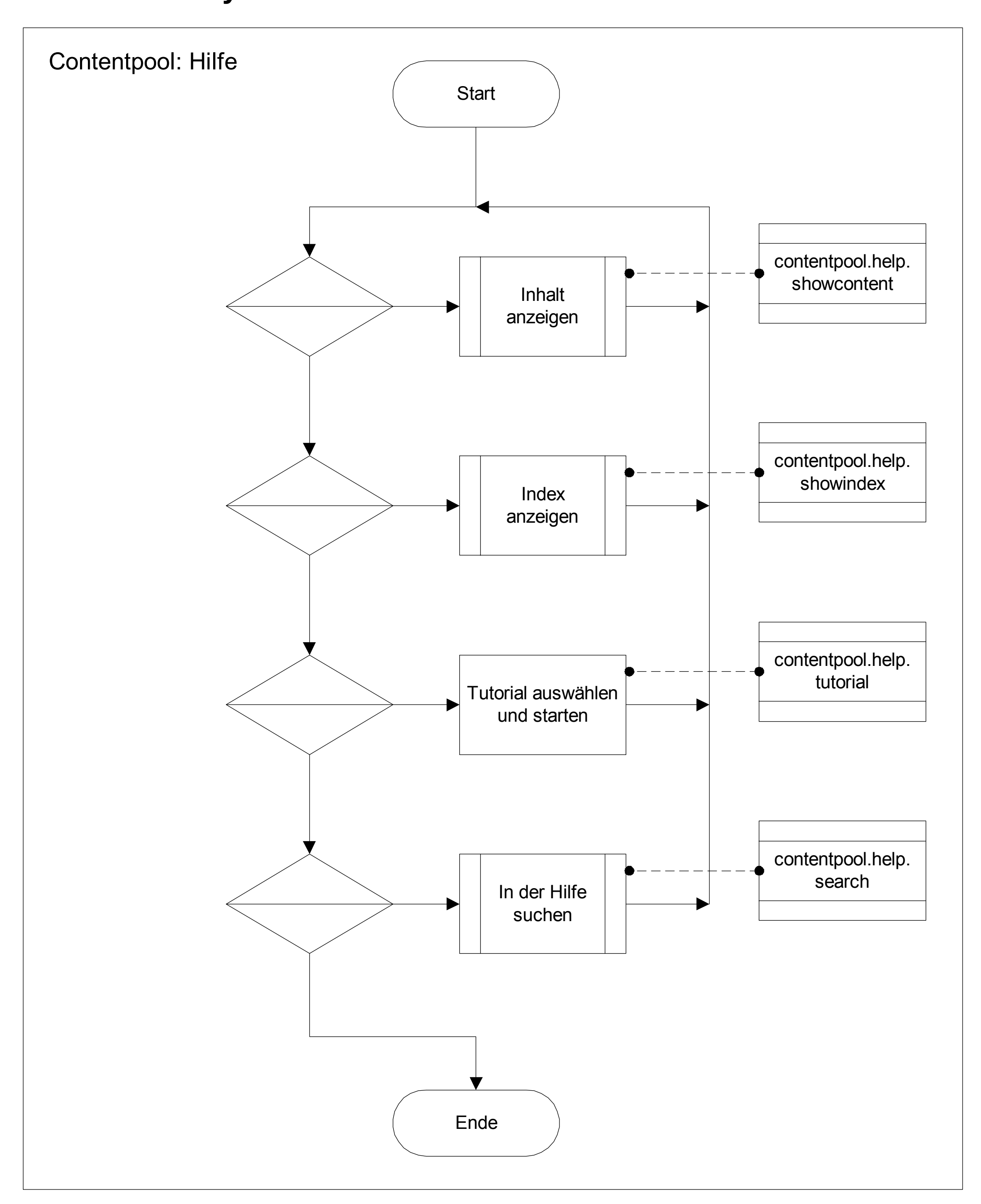


Abbildung 98: Modell des Contentpool-spezifischen Hilfe-Systems

Beschreibung: das Hilfesystem bietet dem Benutzer **Hilfestellungen** zu den Tools und Features der Contentpool-Verwaltung an. Dem Benutzer werden sowohl **Beschreibungen** der Tools und Features als auch **Tutorials**, welche den Umgang mit den zur Verfügung gestellten Werkzeugen beispielhaft demonstrieren, angeboten. Der Zugang zu den Hilfethemen ist mittels eines „herkömmlichen" **Inhaltsverzeichnisses** oder über den nach

Buchstaben organisierten **Index** möglich. Die Hilfe zum Contentpool ist vor allem für ungeübte Benutzer gedacht.

Inhalt anzeigen: in diesem Modus wird zuerst eine Übersicht über die **Kapitel** der Hilfe angeboten. Aus den angezeigten Kapiteln kann entweder ein Kapitel bzw. Unterkapitel ausgewählt oder der Inhalt eines Hilfethemas angezeigt werden. Stellt man sich das Hilfesystem als Baum vor, so sind die Kapitel als die inneren Knoten, die Inhalte hingegen als die Blätter dieses Baums zu betrachten.

Index anzeigen: über den Index hat der Benutzer die Möglichkeit, im Index zu **suchen** (nach einem Wort oder einer Phrase), einen **Indexbuchstaben** oder einen **Unterindex** anzuzeigen. Ist der gewünschte Inhalt gefunden, kann dieser angezeigt werden.

Tutorial starten: es wird zuerst eine Übersicht über die im System vorhandenen Tutorials angezeigt. Aus dieser Liste kann der Benutzer ein gewünschtes Tutorial auswählen und die Präsentation starten.

Suche in der Hilfe: der Benutzer kann ein Wort und eine Phrase eingeben. Anschließend wird nach diesem Schlagwort eine Suche innerhalb der Hilfeeinträge durchgeführt und die Ergebnisse (die Treffer) angezeigt.

In der Folge werden die Prozessmodelle zu den oben angeführten und beschriebenen Features präsentiert:

- 6.2.2.5.1 Index der Hilfe anzeigen .... Seite 217
- 6.2.2.5.2 Im Index der Hilfe suchen .... Seite 218
- 6.2.2.5.3 Inhalt von Hilfethemen anzeigen .... Seite 219

6.2.2.5.1 *Index der Hilfe anzeigen*

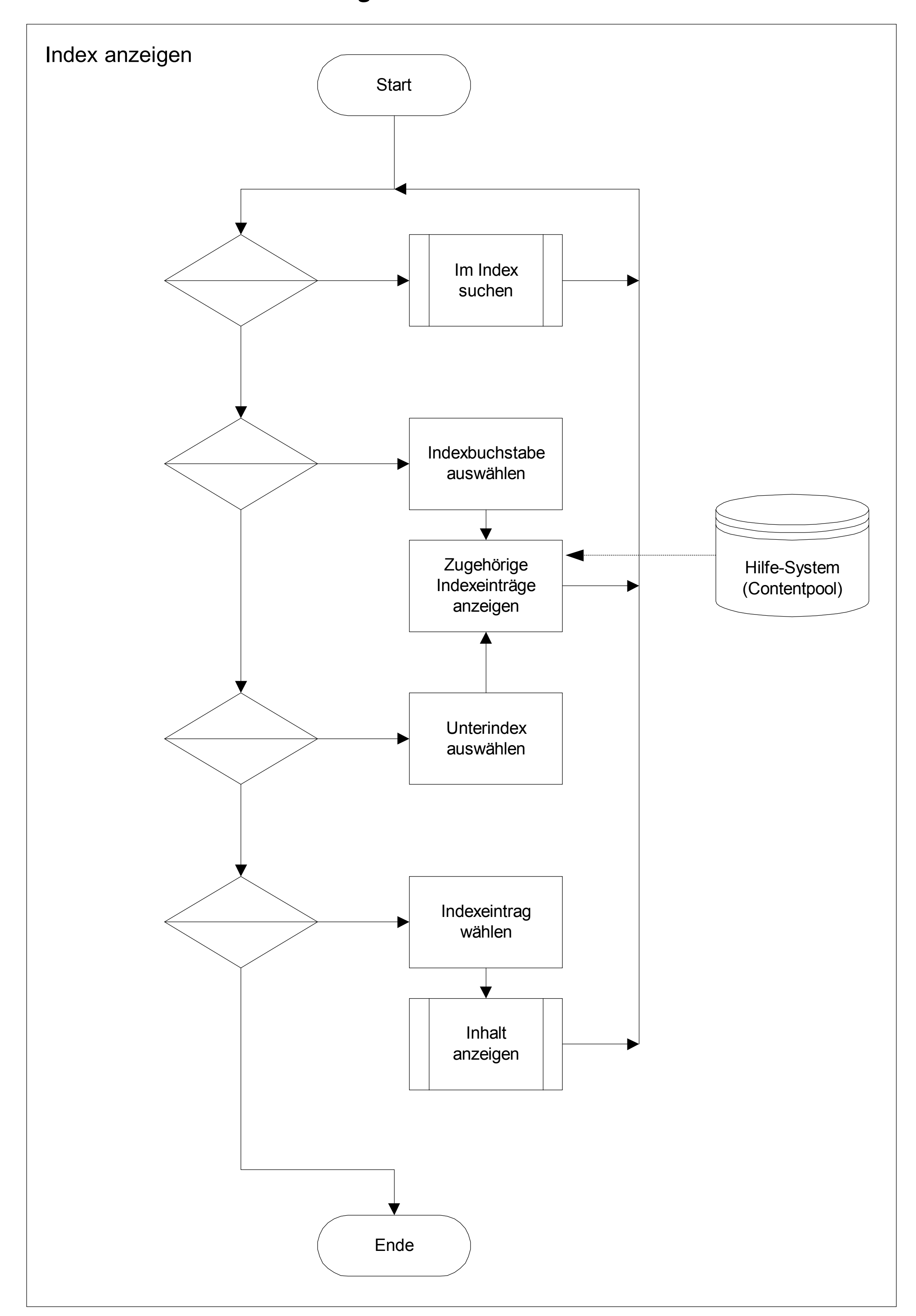


Abbildung 99: Hilfe-System: Index anzeigen

6.2.2.5.2 *Im Index der Hilfe suchen*

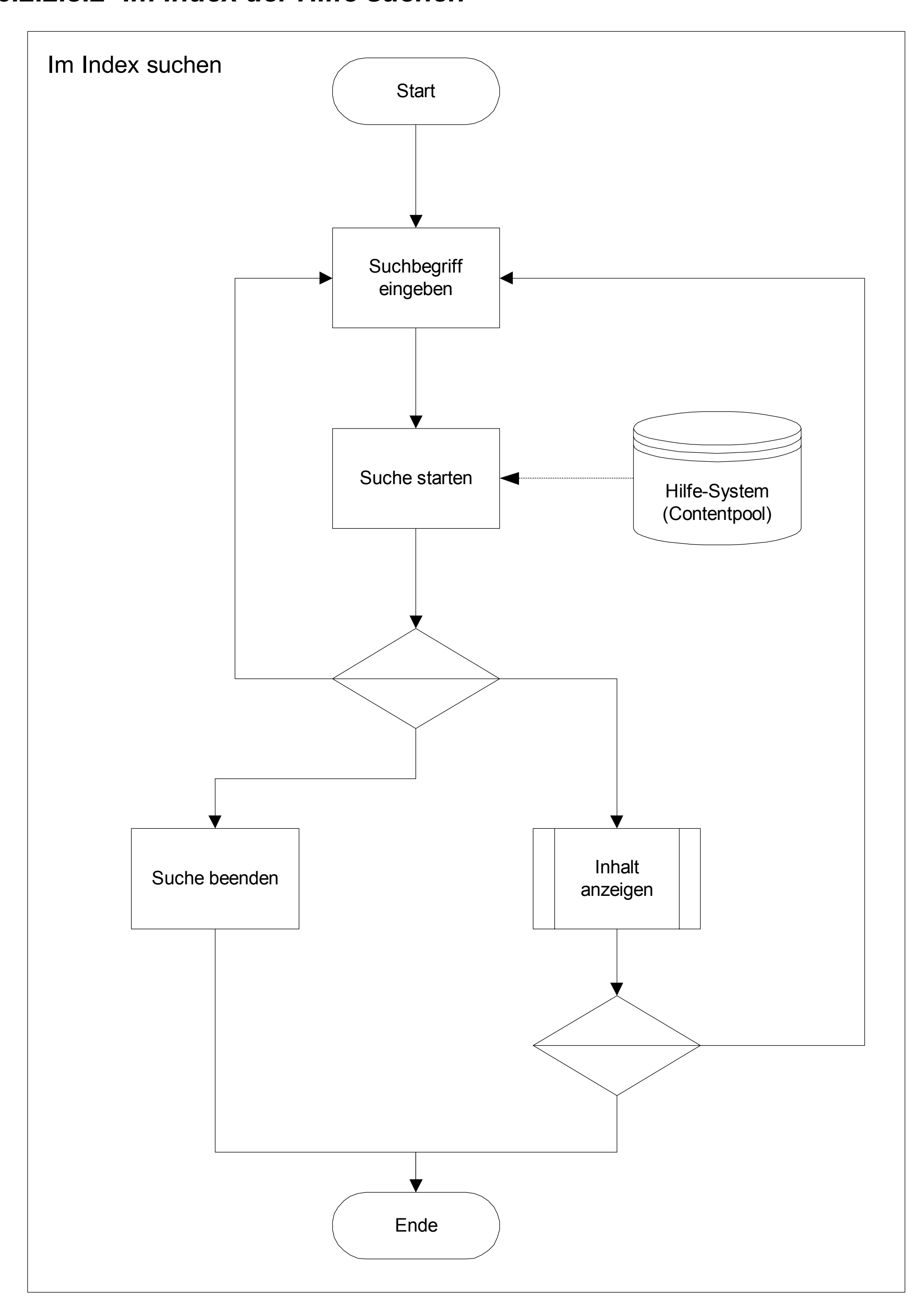


Abbildung 100: Hilfe-System: Suche im Index

6.2.2.5.3 Inhalt von Hilfethemen anzeigen

Abbildung 101: Hilfe-System: Inhalt der Hilfe anzeigen

6.2.3 Systemprozesse

Ziel der Modellierung von Systemprozessen ist, einen konzeptuellen Entwurf von wichtigen, in der Contentpool-Verwaltung verwendeten **Algorithmen** vorzunehmen. Viele dieser Entscheidungen können erst im Verlauf oder zu Beginn der Implementierungs-Phase getroffen werden. Deshalb wird hier noch kein Entwurf der internen Systemprozesse vorgenommen.

An dieser Stelle seien lediglich zusammenfassend alle in den Prozessmodellen spezifizierten Systemprozesse nochmals angeführt. In Klammern ist der Prozess angegeben, in dem der jeweilige Systemprozess spezifiziert und benötigt wird.

- Dateityp des elektronischen Dokuments bestimmen (Zerlegungs-Tool im Dokument Splitter – vgl. Abbildung 66, Seite 179)

- Zerlegungsalgorithmus durchführen (Zerlegungs-Tool im Dokument Splitter – vgl. Abbildung 66, Seite 179)
- Ergebnisse der Zerlegung speichern (Zerlegungs-Tool im Dokument Splitter – vgl. Abbildung 66, Seite 179)
- Extraktion der Metadaten vornehmen (Metadaten-Tool im Dokument Splitter – vgl. Abbildung 69, Seite 182)
- Ergebnisse der Metadaten-Extraktion speichern (Metadaten-Tool im Dokument Splitter – vgl. Abbildung 69, Seite 182)
- Veränderungen an den Metadaten (Versionshistorie) aufzeichnen (Aufbereitungs-Tool im Dokument Splitter – vgl. Abbildung 74, Seite 187; Wissensatom-Verwaltung im Topic-Editor – vgl. Abbildung 84, Seite 198)
- Meta-Metadaten bestimmen (Aufbereitungs-Tool im Dokument Splitter – vgl. Abbildung 74, Seite 187; Wissensatom-Verwaltung im Topic-Editor – vgl. Abbildung 84, Seite 198)
- Technische Daten (z.B. URI, Dateigröße) bestimmen (Aufbereitungs-Tool im Dokument Splitter – vgl. Abbildung 74, Seite 187; Wissensatom-Verwaltung im Topic-Editor – vgl. Abbildung 84, Seite 198)
- Topic Map eines gewählten Fachgebiets sperren (Aufbereitungs-Tool im Dokument Splitter – vgl. Abbildung 75, Seite 188; Wissensatom-Verwaltung im Topic-Editor – vgl. Abbildung 85, Seite 199)
- Topic Map eines gewählten Fachgebiets freigeben (Sperre aufheben) – (Aufbereitungs-Tool im Dokument Splitter – vgl. Abbildung 75, Seite 188; Wissensatom-Verwaltung im Topic-Editor – vgl. Abbildung 85, Seite 199)
- Verknüpfungen gelöschter Wissensatome überprüfen (Wissensatom-Verwaltung im Topic-Editor – vgl. Abbildung 86, Seite 200)

Der vielleicht wichtigste Algorithmus für die Contentpool-Verwaltung ist jener für das Zerlegen von elektronischen Dokumenten. Diesem Algorithmus ist ein eigener Abschnitt im aktuellen Kapitel gewidmet (6.4 Entwurf des Splitter-Algorithmus, Seite 230 ff.).

6.2.4 Berechtigungsbaum für die Contentpool-Verwaltung

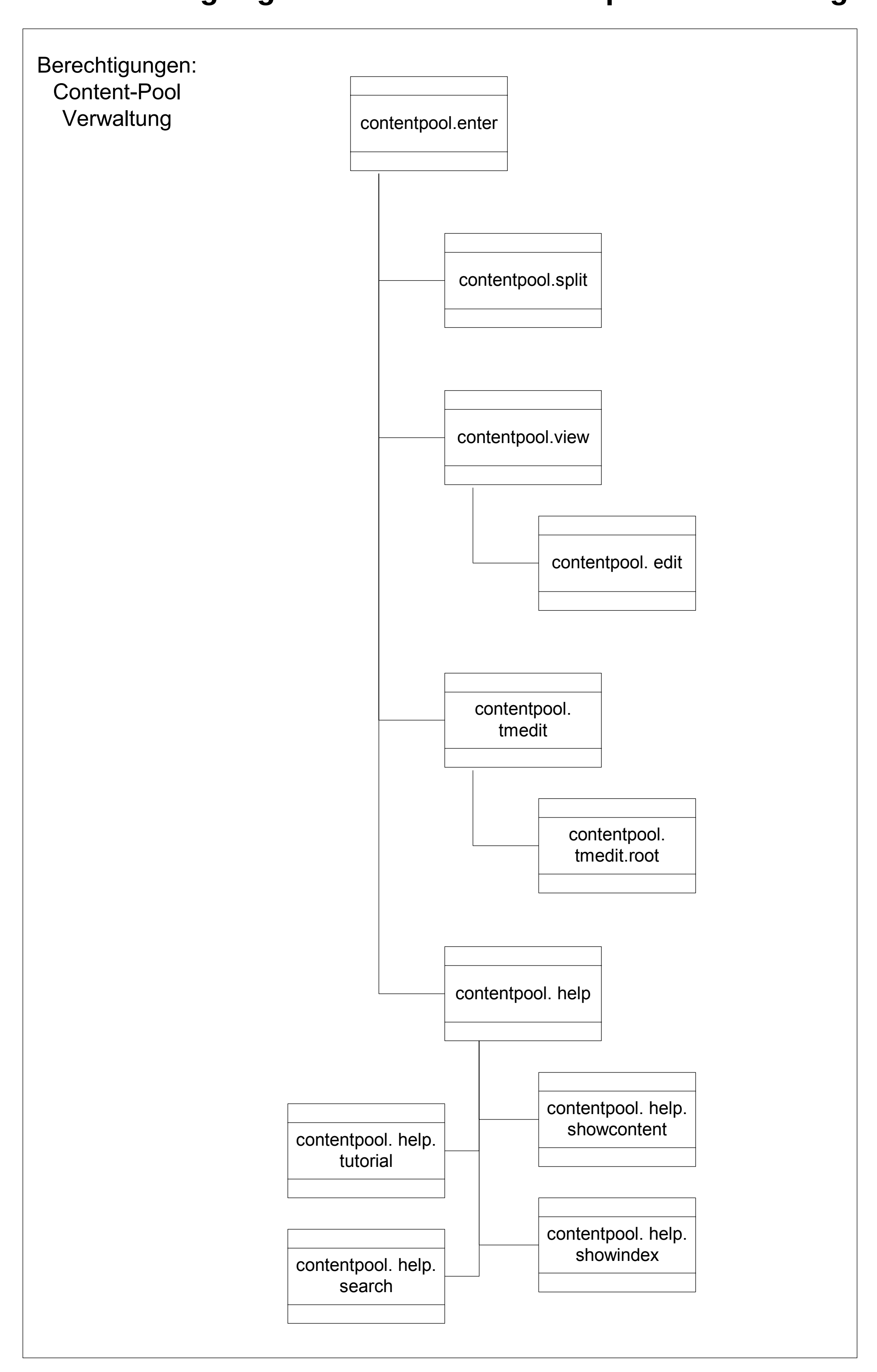


Abbildung 102: Contentpool-Verwaltung – Berechtigungen

Die Berechtigungen für die Contentpool-Verwaltung sind **hierarchisch aufgebaut**, wie aus Abbildung 102 (oben) ersichtlich wird. In den **Blättern** des Berechtigungsbaumes sind die Berechtigungen mit den **meisten Privilegien** symbolisiert; das bedeutet, die Berechtigung „contentpool.tmedit.root“ ist privilegierter als „contentpool.tmedit“; „contentpool.split“ ist privilegierter als „contentpool.enter“, usw.

Wie auch aus der Modellierung der Prozesse ersichtlich wird, erlauben die einzelnen Berechtigungen einem Benutzer, die folgenden Aktionen auszuführen:

- „contentpool.enter“: Der Contentpool darf „betreten“, d.h. die **Contentpool-Verwaltung gestartet** werden. Ansonsten sind keine Privilegien mit dieser Berechtigung verknüpft. Um ein Werkzeug starten zu dürfen, wird eine (oder mehrere) der umfassenderen (d.h. privilegierteren) Berechtigungen benötigt.
- „contentpool.view“: Das **Präsentationstool** darf gestartet werden. Der Benutzer ist also berechtigt, Wissensatome und die dazu gehörenden semantischen Netze mit Hilfe des Topicmap-Navigators zu betrachten.
- „contentpool.edit“: Sowohl das **Präsentationstool** als auch der **Topic-Editor** dürfen gestartet werden. Der Benutzer ist folglich berechtigt, Wissensatome und die dazu gehörenden semantischen Netze mit Hilfe des Topicmap-Navigators zu betrachten sowie mit Hilfe des Topic-Editors neu zu erstellen, zu editieren oder zu löschen.
- „contentpool.tmedit“: Der **Topicmap-Manager** darf gestartet werden. Der Benutzer ist berechtigt, Fachbereiche (Topic Maps) mit Hilfe des Topicmap-Managers neu zu erstellen, zu editieren (bearbeiten) oder zu löschen.
- „contentpool.tmedit.root“: Der Topicmap-Manager darf mit weiteren Privilegien gestartet werden. Zusätzlich zu den oben (vgl. „contentpool.tmedit“) genannten Aktionen darf der Benutzer die **Kategorien-Tabelle** (in welcher die allgemeine Ontologie gespeichert ist, vgl. die Definition auf Seite 207) bearbeiten. Eine Kategorien-Tabelle kann nur einmal existieren; bearbeiten und löschen einer (der) Kategorien-Tabelle ist daher unmöglich.
- „contentpool.help“: die **Contentpool-spezifische Hilfe** von Scholion WB+ darf gestartet werden. Ähnlich wie bei der Berechtigung „contentpool.enter“ sind keine weiteren Privilegien damit verknüpft. Der Benutzer benötigt also eine der weiteren Hilfe-Berechtigungen, um die Contentpool-spezifische Hilfe nutzen zu können.
- „contentpool.help.showcontent“: der Benutzer darf den **Inhalt** der Contentpool-spezifischen **Hilfe** von Scholion WB+ betrachten.
- „contentpool.help.showindex“: der Benutzer darf den **Index** zur Contentpool-spezifischen **Hilfe** von Scholion WB+ nutzen. Im Index sind Stichworte zu den wichtigsten Konzepten der Contentpool-Verwaltung gespeichert, die einen schnelleren Zugriff auf Teile der Hilfe bieten.
- „contentpool.help.tutorial“: der Benutzer darf **Tutorials zu den Werkzeugen** der Scholion WB+ Contentpool-Verwaltung starten und betrachten. Tutorials geben eine kurze Einführung in die Tools, wobei dem Benutzer an Hand von Szenarien die wichtigsten Funktionen der Tools vorgeführt werden.

- „contentpool.help.search“: der Benutzer darf die **Suchfunktion** der Contentpool-spezifischen Hilfe von Scholion WB+ verwenden. Die Suche erlaubt es, alle Einträge der Contentpool-spezifischen Hilfe nach Schlagwörtern zu durchsuchen. Auf diese Weise kann gezielt nach bestimmten Themen nach Hilfe gesucht werden.

6.3 Datenmodell

Zweck dieses Kapitels ist es, einen Überblick über die Modellierung der für die Scholion WB+ Contentpool-Verwaltung benötigten Daten zu geben. Die Spezifikation des Datenmodells erfolgt dabei in zwei Schritten: zuerst wird das konzeptuelle Datenmodell in UML-Notation präsentiert (Abschnitt 6.3.1 Konzeptuelles Datenmodell in UML). Danach erfolgt eine Spezifikation aller Teile der Daten, welche im XML-Format benötigt werden (Abschnitt 6.3.2 XML Schema Definitionen).

6.3.1 Konzeptuelles Datenmodell in UML

In diesem Abschnitt werden aus den Prozessmodellen die **Datenbanktabellen und Beziehungen** entworfen, die für die Speicherung des Contentpools in einem relationalen Datenbanksystem benötigt werden. Die Gesamtheit der Tabellen ergibt das konzeptuelle Datenmodell.

Zur Darstellung des Datenmodells wird die Modellierungssprache UML verwendet. Um die Datenbanktabellen und die Beziehungen zu modellieren, werden **UML Klassendiagramme** verwendet. Namen von Tabellen werden stets mit großem, Namen von Attributen mit kleinem Anfangsbuchstaben notiert. Tabellen- und Attributnamen werden im Text in Kursivschrift notiert.

Die Präsentation des konzeptuellen Datenmodells erfolgt in zwei Teilen:

- Zuerst werden die Tabellen dargestellt, die für die **Modellierung von semantischen Netzen** (auf XML Topic Maps basierend) notwendig sind (siehe Abbildung 103). Die in Abbildung 103 gezeigten Tabellen und Beziehungen bilden all jene Objekte ab, die aus dem XML Topic Map Paradigma (vgl. dazu Abschnitt 4.4.2, Seite 77 ff.) für das Datenmodell des Contentpools berücksichtigt werden.
- Anschließend stehen jene Tabellen im Mittelpunkt, die für die **Modellierung von Kursmaterialien und Nachschlagewerken** notwendig sind (siehe Abbildung 104). Die beiden letztgenannten Objekte sind jene Dokumente, die aus dem Contentpool erzeugt werden. Die in Abbildung 104 modellierten Tabellen sind so entworfen, dass mit geringem Aufwand ein *Content Package* gemäß der IMS Content Packaging Spezifikation (vgl. dazu Abschnitt 4.3.3.2, Seite 63 ff.) erzeugt werden kann.

Hinweis: Die Darstellung des konzeptuellen Datenmodells wurde wie oben beschrieben getrennt, um die Übersichtlichkeit zu erhöhen. Eine Ausnahme von der Einteilung stellt die Tabelle *Cyclopedia* dar (Abbildung 103). Diese dient zur Speicherung von Nachschlagewerken. Da lediglich eine Fremdschlüsselbeziehung zur Tabelle *Atom* besteht, ist die Tabelle *Cyclopedia* in Abbildung 103 dargestellt, obwohl aus sie aus logischen Gründen zu Abbildung 104 gehören würde.

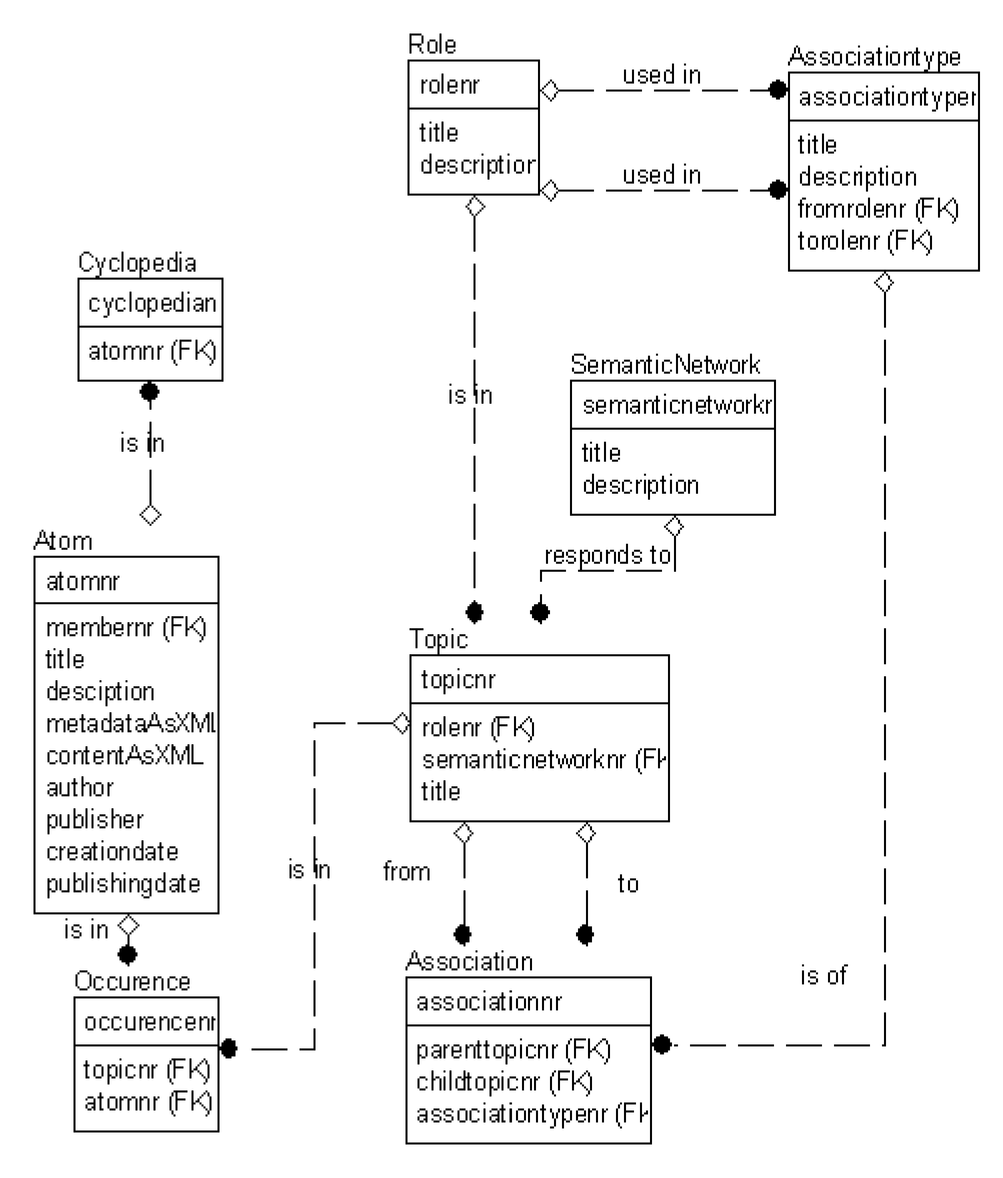


Abbildung 103: Contentpool-Verwaltung – Datenmodell (Teil 1)

Semantische Netze sind in der Tabelle *SemanticNetwork* gespeichert. Sie enthalten jeweils eine beliebig große Anzahl an Topics, die in der Tabelle *Topic* gespeichert sind. Topics sind über Beziehungen mit den Tabellen *Association* und *Occurrence* verbunden. Damit sind die wichtigsten Objekte aus dem Topicmap-Paradigma abgebildet.

Die semantische Bedeutung von Associations wird in den Tabellen *Associationtype* und *Role* modelliert. Im Topicmap-Paradigma ist spezifiziert, dass Topics im Kontext einer Verknüpfung (Association) so genannte Rollen einnehmen dürfen (vgl. [Park, 2002], [Widhalm, 2002], [ISO 13250/2000]). Das konzeptuelle Datenmodell erlaubt die Abbildung von Rollen im Kontext einer Association, indem ein Tupel der Tabelle *Associationtype* die Attribute *fromRoleNr* und *toRoleNr* als Fremdschlüssel der Tabelle Role erhält. Das Attribut *fromRoleNr* definiert die Rolle des Topics, das als Attribut *parentTopicNr* in der Tabelle *Association* eingetragen ist. Das Attribut *toRoleNr* definiert die Rolle des Topics, das als Attribut *childTopicNr* in der Tabelle *Association* eingetragen ist.

Wissensatome sind in der Tabelle *Atom* modelliert und mit den in der Tabelle *Occurrence* gespeicherten Occurrences verknüpft. Diese Vorgehensweise entspricht ebenfalls dem XML Topic Map Standard. Gemäß Standard werden Dokumente stets über Occurrences mit den Topics verknüpft (vgl. [ISO 13250/1999], [XTM, 2001]).

Ein Nachschlagewerk ist in der Tabelle *Cyclopedia* gespeichert und ist definiert als eine Menge aus Wissensatomen in der Tabelle *Atom*.

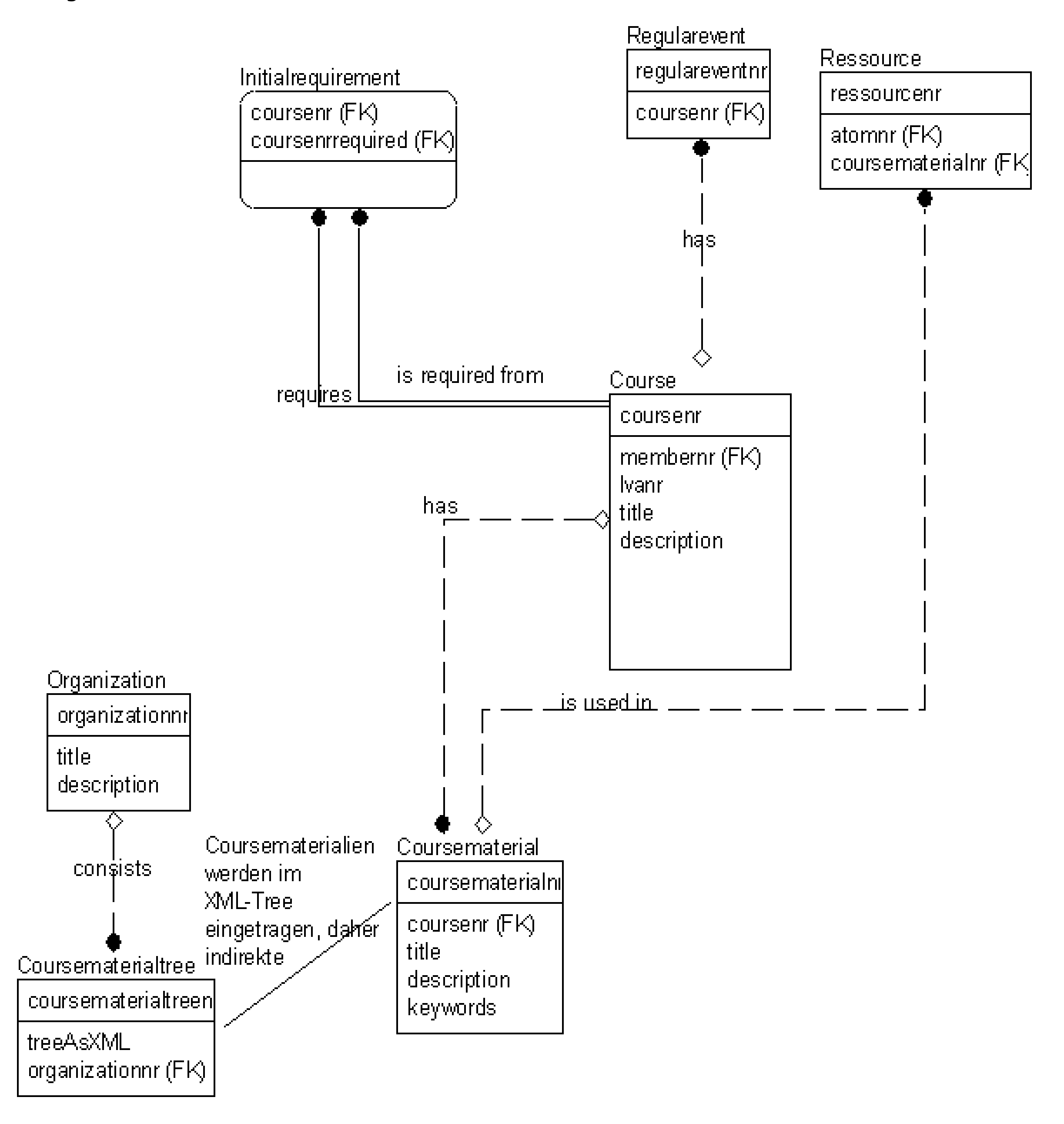


Abbildung 104: Contentpool-Verwaltung – Datenmodell (Teil 2)

In Abbildung 104 wird gezeigt, wie Kursmaterialien modelliert sind. Kursmaterialien sind als Tupel in der Tabelle *Coursematerial* gespeichert. Da zwischen Kursmaterialien und Wissensatomen eine m:n-Beziehung herrscht, wird die Zuordnung eines Wissensatoms

auf ein Kursmaterial über ein Tupel in der Tabelle *Resource* realisiert. Darin sind Wissensatom und Kursmaterial als Fremdschlüssel-Attribute enthalten.

Kursmaterialien sind stets einem Kurs zugeordnet (in der Tabelle *Course* gespeichert). Müssen Voraussetzungen erfüllt werden, um einen Kurs zu absolvieren und das dazu gehörende Kursmaterial zu betrachten, sind Einschränkungen in der Tabelle *Initialrequirement* gespeichert.

Die zu Kursmaterialien übergeordnete Datenstruktur sind Bäume. Solche Bäume können durch Schachtelung von Kursmaterialien entstehen und werden in der Tabelle *Coursematerialtree* gespeichert. Ein Baum aus Kursmaterialien dient zur Modellierung von Zusammenstellungen (z.B. „Literatur-Listen" oder „Skripten" für Kurse) oder auch einer Hierarchisierung (Über- bzw. Unterordnung) von Kursmaterialien. Kursmaterial-Bäume können einem Tupel der Tabelle *Organization* zugeordnet werden.

Die Tabelle *Organization* dient zur Speicherung einer Reihenfolge, welche beim Studium von Kursmaterialien zur Einhaltung empfohlen wird. Dieser Ansatz wurde aus der IMS Content Packaging Spezifikation übernommen, welche eine vergleichbare Vorgehensweise erlaubt (vgl. [IMS Content, 2001]).

6.3.2 XML Schema Definitionen

Ein großer Teil der für die Verwaltung des Contentpools benötigten Daten muss im XML-Format zur Verfügung stehen. Diese Vorgehensweise erleichtert die Einhaltung des **Prinzips der Trennung** von Daten, Verarbeitung und Layout (zu einer Darstellung der Vorzüge einer auf XML basierten Anwendungsarchitektur und einer Darstellung der Gründe für die Entscheidung zur Verwendung dieser Architektur siehe v.a. [Fürlinger, 2003] und [Radmayr, 2003]; vgl. Abschnitt 6.1 Systemarchitektur: Scholion WB+, Seite 143 ff.).

Daraus folgt nicht notwendigerweise, dass die Rohdaten in der Datenbank ebenfalls im XML-Format gespeichert sein müssen. Vielmehr erweist sich meist die Speicherung von Daten als **Attribute einer Tabelle** vorteilhaft gegenüber der Speicherung als XML-Datei. Der Grund dafür liegt darin, dass die Strukturiertheit einer XML-Datei verloren geht, da in der Datenbank nicht die Elemente eines XML-Files, sondern nur das File als Ganzes in Form einer Zeichenkette gespeichert werden kann. Dadurch erleidet man drastische Einbußen bei der Geschwindigkeit der Bearbeitung von Suchanfragen. Zudem können XML-Dateien bei Bedarf effizient aus relationalen Datenbankschemata (d.h. aus Tabellen) generiert werden.

Für die Realisierung der genannten Architektur ist jedoch eine genaue Definition der zu generierenden XML-Daten unerlässlich. In diesem Abschnitt werden daher jene XML-Schema Definitionen vorgestellt, die für den Austausch von Daten zwischen Contentpool und Client-Frontend von Scholion WB+ benötigt werden.

- 6.3.2.1 Kursmaterialstruktur
- 6.3.2.2 Metadaten zu Wissensatomen
- 6.3.2.3 Semantisches Netz (Topic Map)

6.3.2.1 Kursmaterialstruktur

Für die Modellierung der Kursmaterialstruktur in XML wird das vom IMS Global Learning Consortium veröffentlichte **Content Packaging XML Schema** verwendet (vgl. [IMS CP, 2002]). Dieses Schema ist zudem für den Export von Wissensatomen als Grundlage für ein Dateiformat von Interesse.

Die folgenden **einfachen Datentypen** werden im Content Packaging XML Schema definiert:

- schemaType
- schemaversionType
- titleType

Die folgenden **komplexen Datentypen** werden im Content Packaging XML Schema definiert:

- dependencyType
- fileType
- itemType
- manifestType
- metadataType
- organizationType
- organizationsType
- resourceType
- resourcesType

Alle oben genannten Datentypen werden zur Definition der Elemente im Content Packaging XML Schema verwendet.

Weitere Details zum IMS Content Packaging XML Schema wurden bereits im Rahmen der Konzeptanalyse vorgestellt (vgl. Abschnitt 4.3.3.2, Seite 63 ff.). Auf eine wiederholte Darstellung wird verzichtet. Für Details wird zudem auf die Quellen [IMS Content, 2001] und [IMS CP, 2002] verwiesen.

6.3.2.2 Metadaten zu Wissensatomen

Für die Modellierung der zu einem Wissensatom gespeicherten Metadaten in XML wird als Grundlage das vom IMS Global Learning Consortium definierte **Learning Resource Metadata Schema** verwendet (vgl. [IMS MD, 2002]). Dieses Schema ist zudem Grundlage für die zur Speicherung von Metadaten der aus Zerlegung von Dokumenten generierten Wissensatome verwendete Datenstrukturen.

Die folgenden **einfachen Datentypen** werden im Learning Resource Metadata XML Schema definiert:

- formatType

- sizeType
- datetimeType
- idType
- metadataschemeType
- catalogType
- minimumversionType
- maximumversionType

Die folgenden **komplexen Datentypen** werden im Learning Resource Metadata XML Schema definiert:

- aggregationlevelType
- annotationType
- catalogentryType
- centityType
- classificationTye
- contextType
- contributeType
- copyrightandotherrestrictionsType
- costType
- coverageType
- dateType
- descriptionType
- difficultyType
- durationType
- educationalType
- entryType
- generalType
- installationremarksType
- intentedenduseroleType
- interactivitylevelType
- interactivitytypeType
- keywordType
- kindType
- langstringType
- learningresourcetypeType
- lifecycleType
- locationType
- lomType
- metametadataType
- nameType
- otherplatformrequirementsType
- personType
- purposeType
- relationType
- requirementType
- resourceType
- rightsType
- roleType
- semanticdensityType
- sourceType
- statusType
- stringType
- structureType
- taxonType
- taxonpathType
- technicalType

- titleType
- typeType
- typicalagerangeType
- typicallearningtimeType
- valueType
- personType

Alle oben genannten Datentypen werden zur Definition der Elemente im Learning Resource Metadata XML Schema verwendet.

Weitere Details zum IMS Learning Resource Metadata XML Schema wurden bereits im Rahmen der Konzeptanalyse vorgestellt (vgl. Abschnitt 4.3.3.1, Seite 58 ff.). Auf eine wiederholte Darstellung wird verzichtet. Für Details wird zudem auf die Quellen [IMS Meta, 2001], [IMS Meta XML, 2001] und [IMS MD, 2002] verwiesen.

6.3.2.3 Semantisches Netz (Topic Map)

Das semantische Netz oder die semantischen Netze im Contentpool werden auf Grundlage der **XML Topic Maps Spezifikation** [XTM, 2001] modelliert. Jedes der vorhandenen Netze besitzt seine eigene Ontologie und kann mit Hilfe des in diesem Abschnitt vorgestellten Schemas in eine Topic Map im XML Format abgespeichert werden.

Ein Teil dieser Spezifikation ist eine Document Type Definition (DTD), in welcher die grundlegenden Elemente des Topicmap-Paradigmas berücksichtigt sind. Für die Modellierung von semantischen Netzen mit Topic Maps im Contentpool wird die **DTD des Konsortiums TopicMaps.org** verwendet (vgl. [XTM dtd, 2001]).

Im Vergleich zu den sonst verfügbaren Implementierungen des ISO/IEC Topic Map Standards (Ontopia XML Topic Map DTD; Ontopia Linear Topic Map Notation; siehe dazu Abschnitt 4.4.2.4: Umsetzung der Topic Map Spezifikation, Seite 91 ff.) bietet diese DTD den Vorteil, dass sie vom offiziell für den Topic-Map-Standard zuständigen Konsortium veröffentlicht wurde. Daher gelten alle Vorteile, die bei Verwendung eines Standards zu erwarten sind. Die zu erwartenden Vorteile wurden im Abschnitt 4.1.1 (siehe Seite 36 ff.) erläutert.

Folgende **Elemente** werden in der XML Topic Map DTD definiert [XTM dtd, 2001]:

- topicMap
- topic
- instanceOf
- subjectIdentity
- topicRef
- subjectIndicatorRef
- baseName
- baseNameString
- variant
- variantName
- parameters
- occurrence
- resourceRef
- resourceData
- association
- member
- roleSpec
- scope
- mergeMap

Hinweis: Die Reihenfolge der Elemente in der obigen Aufzählung entspricht exakt der Reihenfolge, in welcher die Elemente in der DTD [XTM dtd, 2001] definiert sind.

Die mit den oben aufgezählten Elementen einer XML Topic Map modellierten Objekte einer Topic Map wurden in den Abschnitten 4.4.2.2 (Übersicht über das konzeptionelle Modell der XTM, Seite 80 ff.) und 4.4.2.3 (Konzepte der XTM (concepts), Seite 86 ff.) ausführlich erläutert. Auf eine Wiedergabe weiterer Details wird hier verzichtet. Der interessierte Leser wird auf [XTM dtd, 2001] verwiesen.

6.4 Entwurf des Splitter-Algorithmus

Bereits in den Kapiteln 2 (Ausgangssituation und Problemstellung), 3 (Auswahl der Konzepte und Technologien) und 4 (Related Work) wurde auf die Notwendigkeit eines Algorithmus zur Zerlegung elektronischer Dokumente hingewiesen (vgl. insbesondere die Abschnitte 2.2.1 Zerlegung von Dokumenten, Seite 17 ff., 3.3 Konzepte zur Zerlegung von Dokumenten, Seite 31 ff. und 4.2.1 Slicing Book Technology (SBT), Seite 43 ff.). Als Bezeichnung für den Algorithmus wurde „Splitter-Algorithmus" gewählt.

Durch die **Modellierung der Prozesse** für die Verwaltung des Contentpools (vgl. den Abschnitt 6.2 im vorliegenden Kapitel, Seite 168 ff.) wurde bereits festgelegt, welche **Anforderungen** dieser Algorithmus zu erfüllen hat. An dieser Stelle besteht die Aufgabe darin, aus den definierten Anforderungen einen Algorithmus zu formen, der innerhalb eines Werkzeuges, dem so genannten „Dokument-Splitter", implementiert werden kann. Das vorliegende Kapitel beschäftigt sich mit der Entwicklung des Splitter-Algorithmus.

Bevor mit dem Entwurf des Splitter-Algorithmus begonnen werden kann, ist noch erforderlich, einige **ausgewählte Standard-Lehrbücher** zu analysieren (siehe Abschnitt 6.4.1 Typische Inhalte von Standard-Lehrbüchern). Der Zweck dieses Schritts ist, Erkenntnisse zu gewinnen, welche **Arten von Inhalten** in einem elektronischen Dokument für den verarbeitenden Algorithmus zu erwarten sind.

Als Methode zum Entwurf des Splitter-Algorithmus wird die **schrittweise Verfeinerung** eingesetzt (siehe Abschnitt 6.4.2 Schrittweise Verfeinerung des Algorithmus), wobei eine formlose Prosa-Sprache zur Anwendung kommt. Abschließend wird der gefundene Algorithmus in der Algorithmen-Beschreibungssprache „**Jana**" angegeben (siehe Abschnitt 6.4.3 Splitter-Algorithmus im Ablaufdiagramm).

6.4.1 Typische Inhalte von Standard-Lehrbüchern

Dieser Abschnitt verfolgt den Zweck, **typische Merkmale** von Inhalten aus Standard-Lehrbüchern zu charakterisieren. Dies geschieht hinsichtlich der folgenden drei Aspekte:

- Codalität von Daten (Abschnitt 6.4.1.1)
- Struktur von Inhalten (Abschnitt 6.4.1.2)
- Besondere Problemstellungen für den Algorithmus (Abschnitt 6.4.1.3)

Alle drei genannten Aspekte werden in der Folge beleuchtet.

6.4.1.1 Codalität von Daten

Codalität meint die Art des verwendeten **Formats zur Darstellung von Daten** (für eine präzise Definition vgl. Kapitel 10.2: Glossar, Seite 333 ff.). Folgende Codalitäten sind in elektronischen Dokumenten häufig anzutreffen (vgl. [Weidenmann, 1997a]):

- Text
- Hervorgehobener Text
- Abbildungen bzw. Grafiken
- Animierte Abbildungen
- Hypertext oder Querverweise
- Audiodateien (akustische Elemente)
- Videodateien (animierte visuelle Elemente)

6.4.1.2 Struktur von Inhalten

Unter der Struktur von Inhalten wird in dieser Diplomarbeit sowohl eine Hierarchisierung als auch die **semantische Bedeutung** von Inhalten verschiedener Codalität innerhalb eines elektronischen Dokuments verstanden.

Die Struktur von Inhalten ist immer von dem **Fachgebiet abhängig**, dem ein Dokument angehört. Folgende Arten von Inhalten können z.B. in Dokumenten des Fachgebiets „Mathematik" identifiziert werden (vgl. [Dahn, 2001d]):

- Inhalt an sich („Wissensdaten") – Text
- Theorem
- Korollar
- Formel
- Erklärung
- Merksatz
- Beispiel
- Definition eines Begriffes oder Problems
- Metainformation, welche nicht lesbar ist und/oder von einem Programm zur Anzeige des Dokuments versteckt gehalten, d.h. dem Benutzer nicht angezeigt wird (z.B. Datum der letzten Bearbeitung, Autor des Dokuments)

Die obige Aufzählung erhebt keinen Anspruch auf Vollständigkeit. Wichtig ist die Feststellung, dass das Vokabular für die Struktur von Inhalten **von Fachgebiet zu** Fachgebiet meist **starke Unterscheide** aufweist. Für den Splitter-Algorithmus müssen folglich die Besonderheiten eines Fachgebiets konfigurierbar (d.h. modellierbar) sein.

Der Leser wird hierzu auf die Ausführungen zu Ontologien im Contentpool, Abschnitt 6.1.2: Konzeptuelle Architektur des Contentpools, Seite 147 ff., verwiesen.

6.4.1.3 Besondere Problemstellungen für den Algorithmus

In diesem Abschnitt werden Aspekte berücksichtigt, welche für den Entwurf des Algorithmus relativ schwierig zu lösende Problemstellungen aufwerfen. Anhand der untersuchten Dokumente konnten als Problemfelder identifiziert werden:

- Behandlung von Verzeichnissen (Indices) (siehe Abschnitt 6.4.1.3.1)
- Behandlung von Beschriftungen für Abbildungen o. Ä. (siehe Abschnitt 6.4.1.3.2)
- Darstellung der Struktur eines Dokuments (siehe Abschnitt 6.4.1.3.3)
- Darstellung der semantischen Verknüpfungen (siehe Abschnitt 6.4.1.3.4)

6.4.1.3.1 Behandlung von Verzeichnissen (Indices)

Mit dem Begriff „Verzeichnis" ist eine Indexstruktur gemeint, welche dem Benutzer eines Dokuments (z.B. dem Leser oder dem Autor desselben) einen beschleunigten **Zugriff auf bestimmte Elemente oder Textstellen** im Dokument erlaubt. So existieren in den meisten gedruckten Dokumenten Indexstrukturen für die Kapitel (Inhaltsverzeichnis), für Abbildungen (Abbildungsverzeichnis), für Tabellen (Tabellenverzeichnis), usw.

Solche Verzeichnisse sind zumeist als Tabellen oder in vergleichbarer Form organisiert. Sie enthalten eine große Menge von **Hyperlinks** (Querverweisen) auf andere Stellen im Dokument. Die verwendete Bezeichnung „Hyperlink" soll nicht irreführen, denn oft sind Querverweise lediglich als „**Quasi-Hypertext**" vorhanden. Die Bezeichnung „Quasi-Hypertext" steht für einen Querverweis, der nur scheinbar mit einer Stelle im Dokument verknüpft ist: d.h. die referenzierte Stelle wird zwar im Text der Verknüpfung angezeigt und benannt, doch eine Navigation zur bezeichneten Stelle im Dokument unter Zuhilfenahme des Querverweises, z.B. mittels eines Klicks mit der Maustaste, ist nicht möglich.

Für den Splitter-Algorithmus besteht die Schwierigkeit darin, Verzeichnisse mit Hilfe eines **Parsing-Algorithmus** zu erkennen. In der Regel sind die Verzeichnisse durch die verwendete Formatierung allein (z.B. verwendete Schriftart, Hervorhebungen) nicht von gewöhnlichem Fließtext unterscheidbar.

Als Lösung für das Problem muss eine Implementierung des Splitter-Algorithmus dem Benutzer eine Möglichkeit bieten, die **Position** von Verzeichnissen innerhalb eines zu zerlegenden Dokuments **explizit anzugeben**. Jene Stellen im Dokument, die als Verzeichnis markiert sind, werden von der Zerlegung **in Wissensatome nicht berücksichtigt**; wohl aber für den Schritt der automatischen Erstellung von semantischen Verknüpfungen.

6.4.1.3.2 Behandlung von Beschriftungen für Abbildungen o. Ä.

Abbildungen, Tabellen, Animationen und ähnliche Elemente eines Dokuments sind zumeist mit einer Beschriftung versehen. Diese Beschriftungen könnten vom Splitter-Algorithmus fälschlicherweise als Wissensatom erkannt werden, obwohl sie semantisch lediglich eine **Kurzbeschreibung** des ihnen zugeordneten Elements darstellen und aus diesem Grund den **Metadaten** des zugeordneten Elements zugewiesen werden müssen. Meist sind jedoch Beschriftungen innerhalb eines Dokuments **einheitlich formatiert**

(Schriftart, -größe, -formatierungen, usw.), wodurch die automatische Erkennung mit Hilfe eines Algorithmus erleichtert wird.

Als Lösung für das Problem muss dem Benutzer des Splitter-Algorithmus eine Möglichkeit geboten werden, so genannte „Formatvorlagen" für Beschriftungen im Dokument zu definieren. Alle Stellen im Dokument, welche auf die definierte Formatvorlage passen **und** unmittelbar vor oder nach einer Abbildung bzw. einem entsprechenden ähnlichen Element stehen, werden während des Zerlegungsvorganges nicht als Wissensatome erfasst, sondern automatisch den Metadaten jenes Elements zugeordnet, dessen Beschriftung sie darstellen.

6.4.1.3.3 Darstellung der Struktur eines Dokuments

Der Begriff „Struktur" bezeichnet die **logische Unterteilung** und Gliederung eines Dokuments in Kapitel, Unterkapitel, Abschnitte usw.

Bei der Analyse eines Dokuments kann die Information über dessen logische Struktur mit extrahiert werden. Wenn man die gewonnene strukturelle Information dem Benutzer an der Benutzungsschnittstelle präsentiert, kann seine Arbeit erleichtert werden. Insbesondere die Navigation durch das Dokument kann so erfolgen, wie es dem Benutzer z.B. vom Windows Explorer oder anderen hierarchisch organisierten Programmstrukturen her vertraut ist.

Weiters ist die Struktur des Dokuments für die automatische Gewinnung von Metadaten von Bedeutung (vgl. die Analyse der Slicing Book Technology (SBT) im Abschnitt 4.2.1.2, Seite 46 ff.).

Schwierig gestaltet sich jedoch die Präsentation von Strukturinformation an den Benutzer. Das Werkzeug zur Zerlegung der Dokumente muss die Möglichkeit bieten, einen Baum darzustellen, der die Struktur des Dokuments übersichtlich symbolisiert.

6.4.1.3.4 Darstellung der semantischen Verknüpfungen

Eine wesentliche Anforderung an das Werkzeug zur Zerlegung von Dokumenten ist die Generierung von semantischen Verknüpfungen zwischen den Wissensatomen. Dieser Schritt kann natürlich nur zum Teil automatisch erfolgen.

Dem Benutzer muss eine Möglichkeit geboten werden, die nicht automatisch erstellbaren semantischen Verknüpfungen zu **erstellen**, zu **bearbeiten** und zu **löschen**. Dies stellt eine große Herausforderung für die Gestaltung der Benutzungsschnittstelle des Werkzeugs dar.

6.4.2 Schrittweise Verfeinerung des Algorithmus

In diesem Abschnitt wird aus den eben präsentierten Überlegungen der Algorithmus zum Zerlegen elektronischer Dokumente entwickelt und vorgestellt.

Die verwendete Notation zur Spezifikation von Verbund-Datentypen und Algorithmen orientiert sich an der **Algorithmen-Beschreibungssprache „Jana"** (Java Based Abstract Notation for Algorithms). Jana ist eine Variante der Beschreibungssprache „Jadele" (Java Based Algorithm Description Language), die mit besonders wenigen Konstrukten

auskommt, den Benutzer mit nur wenig syntaktischen Details belastet und dem Benutzer Freiheiten bei der Formulierung überlässt [Jadele, 2000]. Tabelle 10 enthält eine Übersicht der von Jana angebotenen syntaktischen Konstrukte.

Tabelle 10: Syntax der Algorithmen-Beschreibungssprache Jana (nach [Jadele, 2000])

Konstrukt	Ausprägung in Jana
Zuweisungsoperator	`=`
Abschluss von Anweisungen	`; (optional)`
Multiplikation	`*`
Division	`/`
Addition	`+`
Subtraktion	`-`
Anweisung	`x = a / 10;`
Schlüsselwörter	`type, reftype, while, for, if, else, boolean, int, float, char, do, void, static`
Funktionskopf mit Name und Parametern	`squareRoot (↓a ↓ε ↑x)` `int squareRoot (↓a ↓ε)`
Kommentare	`// Endkommentar` `/* Klammerkommentar */`

Hinweise zur Tabelle 10:

- Das **Semikolon** als Zeichen für den Abschluss von Anweisungen ist **optional** definiert. Der Autor verwendet es stets, um die Lesbarkeit des Codes zu verbessern.
- Bei Funktionsköpfen bietet Jana die Möglichkeit, **Ausgangsparameter** (in Tabelle 10 mit dem Symbol ↑ dargestellt) außerhalb der Klammern für Parameter durch die Angabe des Datentyps (ohne einen Namen dafür zu vergeben) zu spezifizieren. Von dieser Möglichkeit macht der Autor Gebrauch. Innerhalb der Klammern des Funktionskopfes werden folglich nur Eingangs- und Übergangsparameter definiert.

In einem ersten Schritt zum Entwurf des Splitter-Algorithmus werden die benötigten Datentypen und Moduln spezifiziert und anschließend der Algorithmus nach der **Methode der schrittweisen Verfeinerung** systematisch entwickelt. Es folgt ein Überblick über die Einzelschritte der Verfeinerung.

Anmerkung: der Algorithmus als Ganzes wird im Detail im Abschnitt 6.4.3 Splitter-Algorithmus im Ablaufdiagramm, Seite 246 ff. präsentiert.

6.4.2.1 Datentypen und Moduln

In diesem Abschnitt werden zuerst Datentypen und Moduln definiert, welche für die Ausführung des Splitter-Algorithmus benötigt werden. Bei den Datentypen wird zwischen den elementaren und den benutzerdefinierten Datentypen unterschieden. Operationen der Datentypen sind nicht im Detail algorithmisch ausformuliert, da es sich in fast allen Fällen um triviale Datenstrukturen und Operationen handelt.

Folgende **elementare Datentypen** werden von Jana angeboten und verwendet (vgl. [Jadele, 2000], Seite 2):

- **Ganze Zahlen** (`int`), deren Wertebereich zwischen minus unendlich und plus unendlich liegt. ([-∞, …, -1, 0, 1, …, ∞])
- **Gleitkommazahlen** (`float`), deren Wertebereich (wie bei int) zwischen minus unendlich und plus unendlich liegt, wobei jedoch sämtliche **Werte zwischen den ganzen Zahlen** erlaubt sind. Der Wertebereich von `float` enspricht also der Menge der reellen Zahlen, $\Re$. ([-∞, …, 0.0, …, ∞])
- **Zeichen** (`char`) beinhalten den gesamten **Zeichenvorrat des ASCII-Codes**. Darin sind sämtliche Buchstaben des lateinischen Alphabets, Sonderzeichen und Steuerzeichen enthalten, wobei die Steuerzeichen im Zusammenhang mit diesem Abschnitt der Diplomarbeit keine Bedeutung besitzen. Variablen vom Datentyp `char` erhalten die Zuweisung von Werten innerhalb von **einfachen Hochkommas** (z.B. `‘a’`, `‘B’`, usw.). **Hinweis**: eine Übersicht zum ASCII-Code findet sich im Glossar (Kapitel 10.2, Seite 333 ff.).
- **Boole’sche Werte** (`boolean`) können nur mit einem von zwei möglichen Symbolen belegt werden: `true` oder `false`. Variablen vom Typ `boolean` enthalten also einen Wahrheitswert.
- **Zeichenketten** (`String`) sind Felder (*Arrays*) von Zeichen (`char`). `String` ist bereits kein elementarer Datentyp mehr, wird aber trotzdem von Jana zur Verfügung gestellt, da Zeichenketten sehr häufig benötigt werden. Der Datentyp `String` besitzt die vordefinierte Funktion `strLen()`, welche die Länge der Zeichenkette liefert. Werte von Zeichenketten werden innerhalb von **doppelten Hochkommas** notiert (z.B. `“abc”`, `“value”`, usw.).

Für den Splitter-Algorithmus werden folgende **benutzerdefinierte Datentypen** verwendet:

- `Atom` definiert die Klasse für Objekte, die zur Speicherung von **Wissensatomen** verwendet werden. Ein Wissensatom besteht aus dem Inhalt in Form einer Zei-

chenkette und aus seinen Metadaten. Der Inhalt wird in einem Objekt der Klasse `Content` gespeichert. Die Metadaten werden in einem Objekt der Klasse `Metadata` „aufbewahrt".

- `Content` ist jene Klasse, die zur Speicherung des **Inhalts von Wissensatomen** dient. Die Schnittstelle der `Content`-Klasse ist sehr einfach: es werden Methoden zum Zugriff auf den Inhalt geboten, welcher in Form einer Zeichenkette von unbegrenzter Länge gespeichert ist.
- `Image` ist eine Subklasse von `Content`, welche zur Speicherung eines **Bildobjekts** verwendet wird. Die Daten des Bildes werden in binärer Form in einem Objekt gespeichert. Objekte der Klasse `Image` bieten Methoden zum Zugriff auf das gespeicherte Bild an.
- `Audio` ist eine Subklasse von `Content`, welche zur Speicherung von **Audio-Inhalten** (Audiodateien) verwendet wird. Die Daten der Audiodatei werden in binärer Form in einem Objekt gespeichert. Objekte der Klasse `Audio` bieten Methoden zum Zugriff auf die gespeicherte Audiodatei an.
- `Metadata` ist eine Klasse für Objekte, welche **Metadaten** zu einem Wissensatom speichern kann. Die Metadaten zum Wissensatom sind so definiert, dass sie dem IEEE LOM Standard entsprechen (vgl. Abschnitt 4.3.2, Seite 54 ff.).
- `AtomQueue` ist eine abstrakte **first-in-first-out Datenstruktur** (Warteschlange, Queue), welche Objekte von Typ `Atom` verwalten kann. Die Klasse `AtomQueue` ist auch in der Lage, dynamisch ihre Aufnahmekapazität an die Anforderungen anzupassen.
- `Document` ist die Klasse für Objekte, welche die **Abstraktion** eines elektronischen Dokuments enthalten (z.B. den Text des Dokuments als Zeichenkette).
- `URI` ist ein Datentyp, der Operationen auf Zeichenketten zur Verwaltung von **Unified Resource Identifiers** (URIs) zur Verfügung stellt. Die Operationen beinhalten insbesondere auch Parsing-Algorithmen, mit deren Hilfe festgestellt werden kann, ob die Zeichenkette eine gültige URI zur Verfügung stellt.
- `Configuration` ist die Klasse für ein Objekt, welches **Einstellungen zur Konfiguration** des Splitter-Algorithmus speichert. Die Einstellungen werden vom Benutzer beim Starten des Splitter-Tools definiert und müssen bis zur vollständigen Ausführung des Splitter-Algorithmus persistent im Speicher gehalten werden. Bei Bedarf kann der Benutzer die Einstellungen zudem dauerhaft auf der Festplatte speichern. Aus diesem Grund wird eine Datenstruktur zur Speicherung der Einstellungen benötigt.
- `Format` ist eine Klasse für Objekte, in denen Daten zu **Formatierungen von Textstellen** gespeichert werden können (z.B. verwendete Schriftart). Objekte vom Typ `Format` werden beim Starten des Splitter-Tools vom Benutzer definiert und zur Laufzeit vom Splitter-Algorithmus benötigt, um bestimmte Stellen im Dokument von besonderer Bedeutung, wie z.B. das Inhaltsverzeichnis oder Beschriftungen von Abbildungen, erkennen zu können.

Die genannten Klassen werden nachfolgend in der Algorithmen-Beschreibungssprache Jana deklariert. Zur Illustration der Klassen wird zusätzlich ein Modell in der Modellierungssprache UML präsentiert (vgl. Abbildung).

```
reftype Atom = {
   Content genericContent;          // May be text, an image, and so on
   Metadata mData;                  // Data describing the content data
   int seqNr;                       // Index number
}

reftype Metadata = {
   int atomNr;                      // Reference to Atom
}

reftype Content = {
   String text;
   Image image;
   Audio audio;
   boolean isText();
   boolean isImage();
   boolean isAudio();
   String getText();
   Image getImage();
   Audio getAudio();
   void setText(String txt);
   void setImage(Image img);
   void setAudio(Audio aud);
}

reftype AtomQueue(↓ int initialSize) = {
   void push(Atom a);               // Add an additional Atom
   Atom pop();                      // Remove least recently added Atom
   int capacity();                  // Get the max. size currently set
   int size();                      // Get number of items stored
   Atom get(int position);          // Get item stored at “position”
}

reftype Document(URI uri) = {
   boolean hasMoreTokens(Granularity g);
```

```
    Content getNextToken(Granularity g);
    boolean hasMoreMetaTokens(Granularity g);
    Metadata getNextMetaToken(Granularity g);
}

reftype URI(String value) = {
    boolean exists();
}
type Granularity = (
    "word",
    "sentence",
    "paragraph",
    "page",
    "chapter"
);

reftype Configuration = {
    // Member variables of a Configuration object
    static Granularity level = "paragraph";
    Format indexFormat;
    Format metadataFormat;

    // Operations allowing access to a Configuration object's member variables
    Format getIndexFormat();
    Format getMetadataFormat();
    Granularity getGranularity();
    void setIndexFormat(↓ Format newFormat);
    void setMetadataFormat(↓ Format newFormat);
    void setGranularity(↓ Granularity newLevel);
}

type Style = (
    "plain",                        // text plain
    "bold",                         // text bold
    "italic",                       // text italic
    "underlined",                   // text underlined
    "bold italic",                  // text bold italic
    "bold ulined",                  // text bold underlined
```

```
    "italic ulined",              // text italic underlined
    "bold it ulined"              // text bold italic underlined
);

reftype Format = {
    String fontType;
    int fontSize;
    Style style;
}
```

6.4.2.2 Oberste Abstraktion (Übersicht) des Algorithmus

Der Algorithmus `splitDocument()` erzeugt aus einem elektronischen Textdokument Wissensatome. Zu Beginn können optional die **Parameter des Algorithmus** durch den Benutzer konfiguriert werden. Verzichtet der Benutzer auf die Konfiguration der Parameter, wird die vom Programm vorgegebene **Standard-Konfiguration** gewählt. Danach werden (in der angegebenen Reihenfolge) Schritte zur Erkennung von Grenzen zwischen den Wissensatomen, Extraktion der Metadaten und manuelle Überarbeitung der generierten Wissensatome durchgeführt.

Grenzen zwischen den Wissensatomen werden gemäß der vom Benutzer gewählten Konfiguration mit Hilfe eines **pattern matching Algorithmus** gesucht und gekennzeichnet. **Metadaten** werden, soweit dies aus dem Dokument möglich ist, extrahiert und den Wissensatomen zugewiesen. Schließlich hat der Benutzer die Möglichkeit, die **Ergebnisse** der automatischen Zerlegung **manuell zu überarbeiten**. Sowohl die Grenzen zwischen den Wissensatomen als auch die zugewiesenen Metadaten können verändert werden.

Nach erfolgreicher Ausführung des Algorithmus werden die erzeugten Wissensatome in einem Array aus Objekten vom Datentyp `Atom` zurückgegeben. Der Algorithmus erwartet keine Eingabeparameter.

```
Atom[] splitDocument()
{
    // Declare all variables needed
    static Configuration CONFIG;
    Document source;
    AtomQueue queue;
    Atom[] results;

    initSplitter();
    source = prepareSplitter();
    findAtomBorders(↓ source ↕ queue);
```

```
  findMetaData(↓ source ↕ queue);
  processResults(↕ queue);
  results = convertResults(↓ queue);

  // Return the readily computed set of “Knowledge Atoms”
  return results;
} // end splitDocument
```

6.4.2.3 Verfeinerung von initSplitter

Die Prozedur `initSplitter()` führt **vorbereitende Arbeiten** durch, die notwendig sind, um mit der Ausführung des Algorithmus beginnen zu können. So werden z.B. globale Variablen initialisiert. Es wird angenommen, dass die im Abschnitt 6.4.2.2 definierten Variablen `source`, `queue` und `results` innerhalb der init-Prozedur sichtbar sind.

```
void initSplitter()
{
  // Choose an initial queue size of 10
  int initialsize = 10;
  // Initialize all variables needed
  source = new Document();
  queue = new AtomQueue(↓ initialsize);
  results = new Atom[initialsize];
  for(int i = 0; i < initialsize; i++)
    results[i] = new Atom();
} // end initSplitter
```

6.4.2.4 Verfeinerung von prepareSplitter

Die Funktion `prepareSplitter` hat die Aufgabe, nach der erfolgten Initialisierung des Algorithmus (vgl. Abschnitt 6.4.2.3 Verfeinerung von initSplitter) **weitere Vorbereitungen** für die Ausführung der Zerlegung eines Dokuments zu treffen. Es werden das zu zerlegende **Dokument ausgewählt** und **Einstellungen des Algorithmus** (seine Konfiguration) bestimmt. Erstmals wird Interaktion mit dem Benutzer benötigt, um die genannten Entscheidungen zu treffen.

```
Document prepareSplitter()
{
  Document doc = new Document();

  doc = openDocument();
  configureSettings();
```

```
    return doc;
  } // end prepareSplitter
```

6.4.2.4.1 Verfeinerung von openDocument

Die Funktion `openDocument` dient zur Auswahl des **zu zerlegenden elektronischen Dokuments**. Als Ausgabeparameter wird das gewählte Dokument innerhalb eines Objekts vom Typ `Document` an den Rufer zurückgegeben; die Funktion erwartet keine Eingabeparameter.

```
Document openDocument()
{
  Document doc;
  URI uri;
  uri = chooseFile();
  if(uri.exists()) doc = new Document(uri);
  else doc = null;

  return doc;
} // end openDocument
```

6.4.2.4.1.1 Abstrakte Operation chooseFile

Mit Hilfe der Operation `chooseFile` kann der Benutzer aus dem lokalen Dateisystem eine **Datei auswählen**, welche in Wissensatome zerlegt werden soll. Die Funktion gibt als Ausgabeparameter ein Objekt des Datentyps `URI` zurück, welches die URI der vom Benutzer gewählten Datei als Zeichenkette enthält und Zugriffsfunktionen darauf zur Verfügung stellt. Es werden keine Eingabeparameter benötigt.

6.4.2.4.2 Verfeinerung von configureSettings

Die Prozedur `configureSettings` hat die Aufgabe, wichtige **Parameter für den Splitter-Algorithmus** zu definieren. Vom Benutzer wird gefordert, dass er die gewünschte Granularität sowie Formatvorlagen für besondere Stellen im Text (z.B. das Inhaltsverzeichnis) definiert. Die Prozedur speichert die vom Benutzer getroffenen Einstellungen in der **statischen Variable** `CONFIG` vom Datentyp `Configuration`. Es werden keine Ein- und Ausgabeparameter erwartet.

```
void configureSettings()
{
  Granularity granLevel;
  String ixFormat;
  String mdFormat;
```

```
    CONFIG = new Configuration();

    granLevel = chooseGranularity();
    ixFormat = chooseIndexFormat();
    mdFormat = chooseMetadataFormat();

    CONFIG.setGranularity(granLevel);
    CONFIG.setIndexFormat(ixFormat);
    CONFIG.setMetadataFormat(mdFormat);
} // end configureSettings
```

6.4.2.4.2.1 Abstrakte Operation chooseGranularity

Mit Hilfe der Operation `chooseGranularity` kann der Benutzer aus einer Menge von vorgegebenen Einstellungen die **gewünschte Größeneinheit eines Wissensatoms** konfigurieren. Die Menge der vorgegebenen Einstellungen orientiert sich am **Aufzählungsdatentyp `Granularity`** (vgl. Abschnitt 6.4.2.1 Datentypen und Moduln). Die Funktion gibt als Ausgabeparameter ein Objekt des Aufzählungsdatentyps `Granularity` zurück, welches in einem Objekt des Datentyps `Configuration` zurück gespeichert wird. Es werden keine Eingabeparameter benötigt.

6.4.2.4.2.2 Abstrakte Operation chooseIndexFormat

Mit Hilfe der Operation `chooseIndexFormat` legt der Benutzer eine **Formatvorlage für Verzeichnisse** (z.B. das Inhaltsverzeichnis) im elektronischen Quelldokument fest. Die Funktion gibt als Ausgabeparameter ein Objekt des Datentyps `Format` zurück. Im Rückgabeobjekt ist Information über das Schriftbild von Indexstrukturen im Dokument gespeichert, wie z.B. verwendete Schriftart, Schriftgröße und Hervorhebungen. Die Operation erwartet keine Eingabeparameter.

6.4.2.4.2.3 Abstrakte Operation chooseMetadataFormat

Mit Hilfe der Operation `chooseMetadataFormat` definiert der Benutzer eine Formatvorlage für Textstellen im elektronischen Dokument, welche für **Beschriftungen von Dokument-Elementen** (z.B. Abbildungen) verwendet wird. Gefundene Beschriftungen sind nicht als eigene Wissensatome, sondern als Metadaten der ihnen zugeordneten Dokument-Elemente zu betrachten. Die vom Benutzer definierten Einstellungen werden als Ausgabeparameter in einem Objekt vom Datentyp Format an den Rufer zurückgegeben. Im Rückgabeobjekt ist Information über das Schriftbild von Indexstrukturen im Dokument gespeichert, wie z.B. verwendete Schriftart, Schriftgröße und Hervorhebungen. Die Operation erwartet keine Eingabeparameter.

6.4.2.5 Verfeinerung von findAtomBorders

Die Funktion `findAtomBorders` hat als Aufgabe, die (vorläufigen) **Grenzen zwischen den Wissensatomen** im elektronischen Quelldokument zu finden. Jene Stellen, an de-

nen der Text des Dokuments zerteilt werden soll, werden mit Hilfe der vom Benutzer definierten Granularität (vgl. Abschnitt 6.4.2.4.2 Verfeinerung von configureSettings, Punkt 6.4.2.4.2.1 Abstrakte Operation chooseGranularity) nach einem **Mustersuchalgorithmus** (pattern search) aufgespürt und gekennzeichnet.

Als Übergangsparameter wird der Funktion ein Objekt vom Datentyp `AtomQueue` übergeben. Die `AtomQueue` enthält anfangs keine Objekte; während der Zerlegung wird sie mit Objekten des Typs `Atom` befüllt. Jedes `Atom`-Objekt enthält ein Wissensatom, welches als Inhalt ein jeweils von den gesuchten Trennzeichen begrenztes Bruchstück des Quelldokuments enthält. Als zusätzlicher Eingabeparameter wird ein Objekt vom Typ `Document` erwartet, worin das **elektronische Quelldokument** enthalten ist und welches Zugriffsfunktionen auf den Inhalt des Dokuments zur Verfügung stellt. Außer `queue` werden keine weiteren Ausgangsparameter benötigt.

```
findAtomBorders(↓ Document source ↕ AtomQueue queue)
{
  Atom atom;
  Content cont;

  Granularity granularity = CONFIG.getGranularity();
  while(source.hasMoreTokens(granularity))
  {
    cont = source.getNextToken(granularity);
    atom = new Atom();
    atom.setContent(cont);
    queue.push(atom);
  } // end while
} // end findAtomBorders
```

6.4.2.6 Verfeinerung von findMetaData

Die Funktion `findMetaData` hat die Aufgabe, alle **Textstellen** im elektronischen Quelldokument **zu finden**, welche auf die zuvor vom Benutzer definierte Formatvorlage für Beschriftungen passen. Gefundene Metadaten-Textstellen werden vom Algorithmus dem entsprechenden Wissensatom zugeordnet.

Als Übergangsparameter wird der Funktion ein Objekt vom Datentyp `AtomQueue` übergeben. Die `AtomQueue` enthält die Wissensatome, die zuvor durch die Zerlegung des elektronischen Quelldokuments gefunden werden konnten (vgl. Abschnitt 6.4.2.5 Verfeinerung von findAtomBorders). Als zusätzlicher Eingabeparameter wird ein Objekt vom Typ `Document` erwartet, worin das elektronische Quelldokument enthalten ist und welches Zugriffsfunktionen auf den Inhalt des Dokuments zur Verfügung stellt. Außer `queue` werden keine weiteren Ausgangsparameter benötigt.

```
findMetaData(↓ Document source ↕ AtomQueue queue)
{
  Atom currAtom;
  Metadata meta;

  Granularity granularity = CONFIG.getGranularity();
  while(source.hasMoreMetaTokens(granularity))
  {
    int atomIndex = 0;
    meta = source.getNextMetaToken(granularity);
    atomIndex = meta.atomNr();
    currAtom = queue.get(atomIndex);
    currAtom.mData = meta;
  } // end while
} // end findMetaData
```

6.4.2.7 Verfeinerung von processResults

Mit Hilfe der Funktion `processResults` werden die zuvor erzeugten Wissensatome (vgl. Abschnitte 6.4.2.5 Verfeinerung von findAtomBorders und 6.4.2.6 Verfeinerung von findMetaData) einschließlich der Metadaten bearbeitet. Der Benutzer hat die Möglichkeit, **Grenzen zwischen den Wissensatomen** zu verändern (hinzufügen, verschieben, löschen – d.h. zu große Wissensatome zerteilen, unpassende Zerteilungen verändern, zu kleine Wissensatome verschmelzen) sowie die **Metadaten der Wissensatome** zu bearbeiten (neu einfügen, bearbeiten, löschen).

Als Übergangsparameter muss der Funktion ein Objekt des Datentyps `AtomQueue` zur übergeben werden. Die in der `AtomQueue` gespeicherten Wissensatome werden verändert, falls der Benutzer die oben beschriebenen Änderungen durchführt. Weitere Eingangs- oder Ausgangsparameter werden nicht benötigt.

```
processResults(↕ AtomQueue queue)
{
  processAtomBorders();
  processMetadata();
} // end processResults
```

6.4.2.7.1 Abstrakte Operation processAtomBorders

Mit Hilfe der Operation `processAtomBorders` können die automatisch generierten Grenzen der Wissensatome (vgl. Abschnitt 6.4.2.5 Verfeinerung von findAtomBorders) vom

Benutzer verändert werden. Folgende Vorgänge werden mit Hilfe einer grafischen Benutzungsschnittstelle unterstützt:

- **Hinzufügen** von neuen Wissensatom-Grenzen zum Zerteilen von zu großen, d.h. semantisch „überladenen" Wissensatomen.
- **Verschieben** von generierten Wissensatom-Grenzen zum Korrigieren semantisch unpassender oder unschlüssiger Zerlegungen.
- **Löschen** von generierten Wissensatom-Grenzen zum Verschmelzen von zu kleinen, d.h. semantisch unvollständigen Wissensatomen.

6.4.2.7.2 Abstrakte Operation processMetadata

Mit Hilfe der Operation `processMetadata` können die automatisch zu den Wissensatomen generierten Metadaten (vgl. Abschnitt 6.4.2.6 Verfeinerung von findMetaData) vom Benutzer bearbeitet werden. Folgende Vorgänge werden mit Hilfe einer grafischen Benutzungsschnittstelle unterstützt:

- **Hinzufügen** von Metadaten-Feldern, welche nicht durch eine automatische Erkennung ermittelt werden konnten.
- **Bearbeiten** von Metadaten-Feldern, um unvollständige oder unrichtige Einträge zu ergänzen bzw. korrigieren.
- **Löschen** von Metadaten-Feldern, welche dem Benutzer überflüssig erscheinen.

6.4.2.8 Verfeinerung von convertResults

Die Funktion `convertResults` arbeitet die **Warteschlange** ab, in der die **fertig bearbeiteten Wissensatome** nach Durchführung der automatischen Zerlegung, der automatischen Metadaten-Generierung und der manuellen Überarbeitung der Zerlegung und Metadaten durch den Benutzer gespeichert sind (vgl. dazu die Abschnitte 6.4.2.5 Verfeinerung von findAtomBorders, 6.4.2.6 Verfeinerung von findMetaData und 6.4.2.7 Verfeinerung von processResults). Aus der Warteschlange wird ein **Array erzeugt**, dessen Länge mit der Anzahl der gespeicherten Wissensatome übereinstimmt.

Als Eingangsparameter erwartet die Funktion ein Objekt des Datentyps AtomQueue. An den Rufer wird ein Array aus Objekten des Datentyps Atom als Ausgangsparameter zurückgegeben.

```
Atom[] convertResults(↓ AtomQueue queue)
{
  int size = queue.size();
  Atom[] result = new Atom[size];
  for(int i = 0; i < size; i++)
  {
    result[i] = queue.pop();
  } // end for
} // end convertResults
```

6.4.3 Splitter-Algorithmus im Ablaufdiagramm

Im Abschnitt 6.4.2 Schrittweise Verfeinerung des Algorithmus (Seite 233 ff.) wurden die **Bausteine** des Splitter-Algorithmus bis auf die Ebene von Methoden, Prozeduren und abstrakten Operationen zerlegt. Hier werden die Bausteine zu einem Ganzen zusammen gefügt und somit der Algorithmus in seiner **Gesamtheit** wiedergegeben. Um den Ablauf des Algorithmus besser veranschaulichen zu können, wird das darstellerische Mittel des Ablaufdiagramms gewählt.

Hinweise zum Ablaufdiagramm in Abbildung 105:

- Da der Algorithmus im Ablaufdiagramm vollständig gezeigt wurde, ergeben sich durch die Auflösung von Untermethoden leichte Änderungen im Code gegenüber den durch die schrittweise Verfeinerung gefundenen und formulierten Methoden.
- Um die Übersichtlichkeit des Diagramms zu erhalten, wurden die Methoden und abstrakten Operationen `openDocument()`, `Configuration.init()` und `AtomQueue.init()` nicht im Detail ausformuliert. In den entsprechenden Abschnitten weiter vorne in diesem Dokument können die Spezifikationen dieser Operationen nachgeschlagen werden.

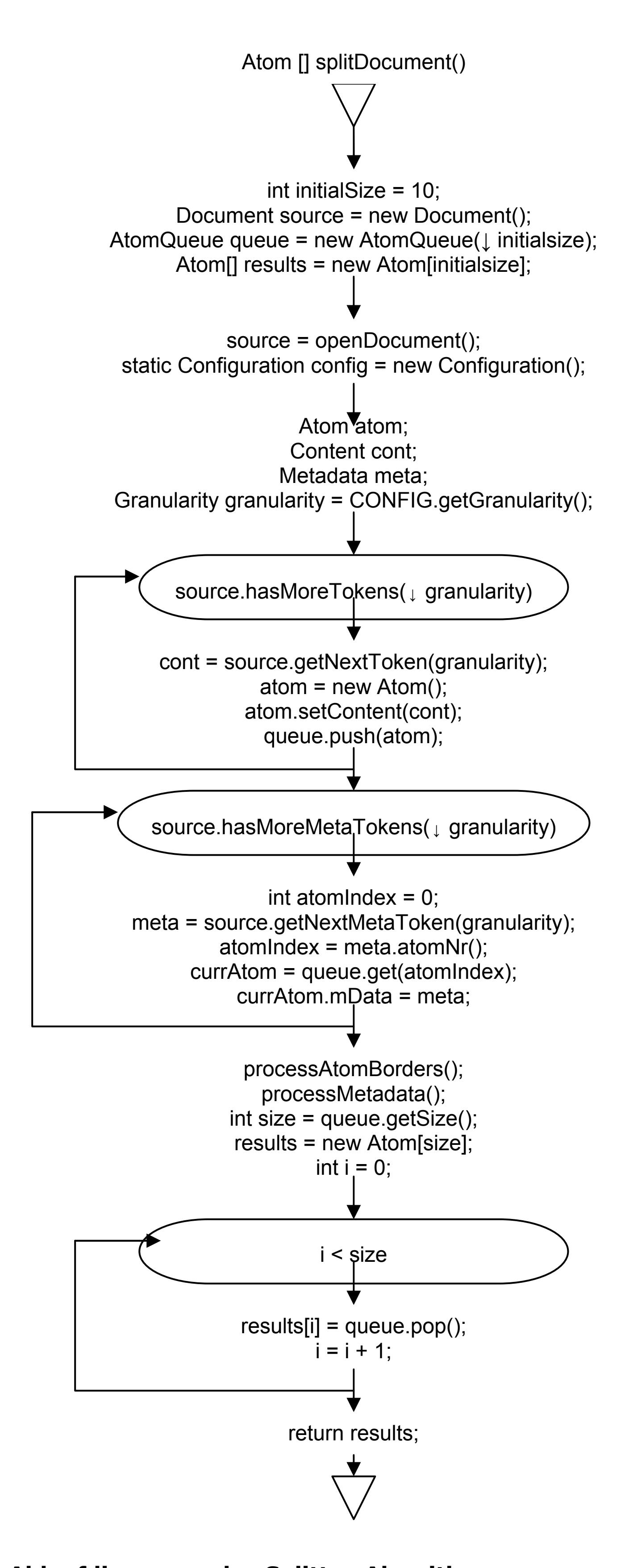


Abbildung 105: Ablaufdiagramm des Splitter-Algorithmus

6.5 Zusammenfassung

Im Kapitel 6 wurden das Design des Contentpools und der Werkzeuge für den Zugriff auf den Contentpool definiert. Mit diesem Schritt wurden die Fragestellungen der Diplomarbeit auf konzeptueller Ebene beantwortet:

- Die Anforderungen für das Werkzeug zur Zerlegung von Dokumenten wurden definiert. Ein Algorithmus, der die Zerlegung realisiert, wurde entwickelt.
- Es wurde ein Datenmodell spezifiziert, in dem die Datenbasis zur Speicherung von Wissensatomen und das semantische Netz zur Speicherung von semantischen Zusammenhängen berücksichtigt wurden.
- Für die Darstellung der Wissensatome wurden die Anforderungen entwickelt. Die Gestaltung der Benutzungsschnittstelle zur Darstellung ist im Rahmen der Implementierung noch zu klären.
- Um die Verwendung von Wissensatomen in neuen Dokumenten zu ermöglichen, wurde eine Suchfunktion im Contentpool spezifiziert.

7 Implementierung der Software

In diesem Kapitel werden die Ergebnisse der Implementierung gemäß des in Kapitel 6 (Design der Software, Seite 143 ff.) spezifizierten Designs vorgestellt. Dabei geht der Autor nicht auf Details der Implementierung ein, da die vollständige Wiedergabe der Dokumentation des API (Application Programming Interface) oder sogar des Quellcodes der Klassen den Rahmen der Diplomarbeit sprengen würde. Die Vorstellung der Implementierungs-Ergebnisse erfolgt vielmehr an Hand von **Klassenmodellen**, aus denen die Umsetzung der im Design getroffenen Entscheidungen im produktiven Programm ersichtlich wird.

Die Implementierung der Werkzeuge für die Contentpool Verwaltung und den Dokument Splitter erfolgte mit Hilfe der **Programmiersprache Java**. Diese Sprache ist besonders gut zur Erstellung webbasierter Applikationen geeignet. Die Verwendung von Java war eine Bedingung für den Start des Projekts „Scholion WB+“ (vgl. dazu auch den Punkt 1.2.2.8 Implementierung der Werkzeuge im Abschnitt 1.1 Ziele dieser Diplomarbeit auf Seite 11). Zum Verständnis des vorliegenden Kapitels werden daher Grundkenntnisse über die Programmiersprache Java vorausgesetzt.

Zur Erhöhung der Übersichtlichkeit werden die Klassenmodelle in zwei Teilen präsentiert:

7.1 Klassenmodell der Contentpool Verwaltung

Die Contentpool-Verwaltung ist ein integraler Bestandteil der Wissenstransfer-Umgebung Scholion WB+ (wie dies in der Architektur spezifiziert wurde; vgl. dazu Abschnitt 6.1 Systemarchitektur: Scholion WB+, Seite 143 ff.). Eine **Abgrenzung** zu den anderen Teilen von Scholion WB+ muss daher zu Beginn erfolgen, noch bevor detailliert auf das Klassenmodell der Contentpool-Verwaltung per se eingegangen werden kann.

Der vorliegende Abschnitt wird daher in folgenden Teil-Abschnitten präsentiert:

7.1.1 Übersicht: Klassenmodell von Scholion WB+

Zu Beginn wird, um die **Einordnung** der Contentpool-Verwaltung in das Projekt Scholion WB+ darzustellen, das Klassenmodell der Implementierung von Scholion WB+ präsentiert. Das Modell zeigt, wie das in Abbildung 53 (Gesamtkonzept für die Systemarchitektur – Scholion WB+, siehe Seite 156) präsentierte konzeptuelle Modell umgesetzt wurde (siehe Abbildung 106).

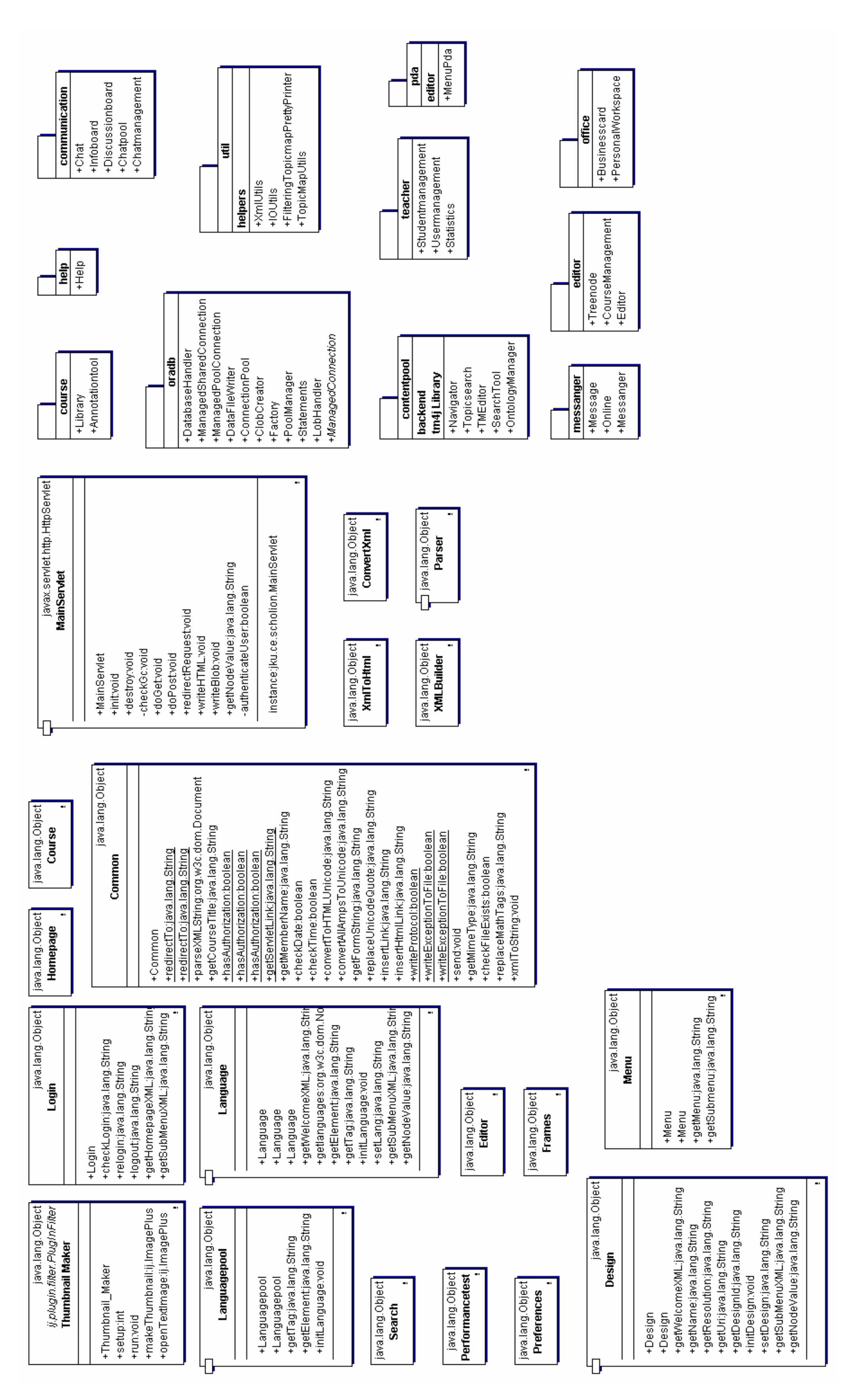


Abbildung 106: Klassenmodell von Scholion WB+ (teilweise vereinfacht)

Alle Klassen der Wissenstransfer-Umgebung Scholion WB+ wurden in einem Package namens `jku.ce.scholion` implementiert. Aus Gründen der leichteren Lesbarkeit wird aber auf die vollständige Angabe dieses Package-Namens im gesamten Abschnitt 7.1 verzichtet. Statt dessen verwendet der Autor stets die kurze Bezeichnung „`scholion`". Selbstverständlich befinden sich auch alle Sub-Packages von `scholion` innerhalb der Packagehierarchie `jku.ce.scholion`.

In der Abbildung 106 wurden einige Klassen, die im Kontext dieses Kapitels für das Verständnis nicht von oberster Priorität sind, **vereinfacht dargestellt**. Dies bedeutet, dass auf die detaillierte Darstellung der Methoden dieser Klassen verzichtet wurde. Der überwiegende Teil der in Abbildung 106 gezeigten Methoden in den Klassen von Scholion WB+ ist zudem für die Darstellung der Implementierung der Contentpool-Verwaltung nicht relevant. Auf eine detailliertere Erläuterung wird daher verzichtet.

In der Folge werden aus der eben gezeigten Architektur von Scholion WB+ die Klassenmodelle der Packages `scholion.contentpool` und `scholion.util` im Detail präsentiert. Darüber hinaus wird das Klassenmodell des Packages `utils.topicmaps` vorgestellt.

7.1.2 Package „scholion.contentpool"

Das Package `scholion.contentpool` verfolgt den Zweck, **Algorithmen zur Verwaltung** von **Topic Maps** und zur Verwaltung von **Wissensatomen** in Scholion WB+ einzubinden. Es besteht aus den folgenden, in den in Klammern genannten Abschnitten beschriebenen Teilen:

- Klasse Navigator (Abschnitt 7.1.2.1)
- Klasse TMEditor (Abschnitt 7.1.2.2)
- Klasse OntologyManager (Abschnitt 7.1.2.3)
- Klasse SearchTool (Abschnitt 7.1.2.4)
- Klasse Topicsearch (Abschnitt 7.1.2.5)
- Sub-Package „backend" (Abschnitt 7.1.2.6)

Abbildung 107 gibt einen Überblick über die oben genannten Teile des Packages `scholion.contentpool`. (**Hinweis**: in der Abbildung wurden mehrere Methoden in den Klassen `Navigator` und `TMEditor` verborgen, um die Darstellung nicht zu unübersichtlich werden zu lassen.)

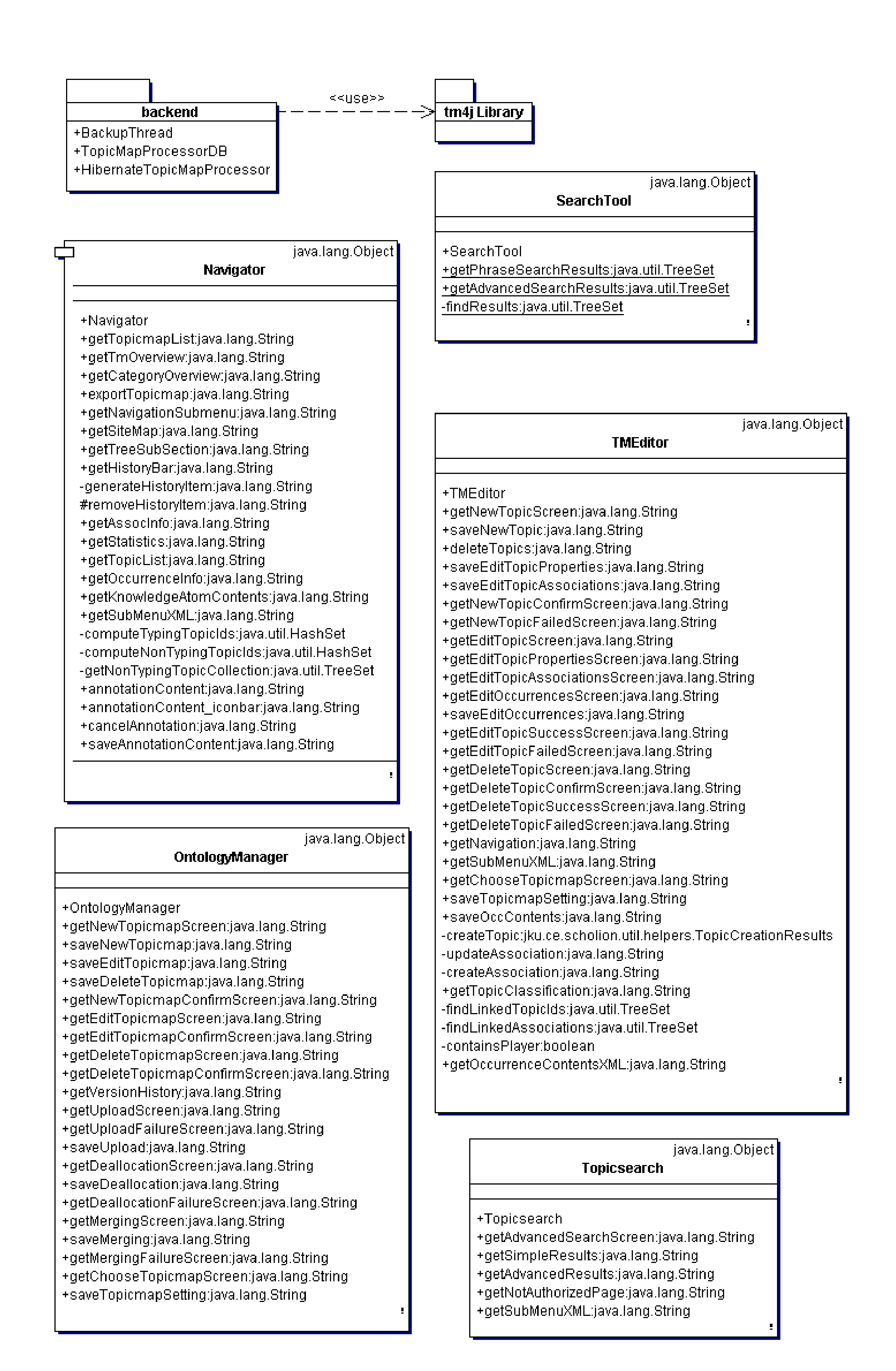


Abbildung 107: Klassenmodell – Package scholion.contentpool

7.1.2.1 Klasse Navigator

`Navigator` dient zum **lesenden Zugriff** auf die Inhalte in einer Topic Map. Der Zugriff geschieht über eine **grafische Benutzungsschnittstelle**, deren äußeres Erscheinungsbild im Wesentlichen mit Hilfe mehrerer Klassen aus dem Package `utils.topicmaps` (vgl. dazu den Abschnitt 7.1.4 in diesem Kapitel, Seite 264 ff.) bestimmt wird.

Folgende Features werden von der Klasse `Navigator` zur Verfügung gestellt (vgl. Abbildung 107):

- Liste der verfügbaren Topic Maps ausgeben
- Übersicht über die Inhalte einer Topic Map anzeigen
- Liste aller verfügbaren Topics in einer Topic Map anzeigen
- Statistische Auswertungen zu einer Topic Map berechnen
- Eine Topic Map als Datei im xtm-Format (vgl. [XTM dtd, 2001]) exportieren
- Einzelne Topics, deren Associations und Occurrences anzeigen

Wie aus der Beschreibung der Funktionalität hervorgeht, können mit Hilfe der Klasse `Navigator` Inhalte aus Topic Maps lediglich gelesen, nicht aber verändert werden.

7.1.2.2 Klasse TMEditor

`TMEditor` bietet Methoden zum **Zugriff auf die Inhalte** in einer Topic Map sowie zum Erstellen, Verändern (Bearbeiten) und Löschen dieser Inhalte. Der Zugriff geschieht, wie bei `Navigator`, über eine **grafische Benutzungsschnittstelle**, deren äußeres Erscheinungsbild im Wesentlichen mit Hilfe mehrerer Klassen aus dem Package `utils.topicmaps` (vgl. dazu den Abschnitt 7.1.4 in diesem Kapitel, Seite 264 ff.) bestimmt wird.

`TMEditor` stellt folgende Features zum Lesen und Schreiben von Inhalten einer Topic Map zur Verfügung (vgl. Abbildung 107):

- Zu bearbeitende Topic Map auswählen
- Erstellen, Bearbeiten und Löschen von Topics
- Erstellen, Bearbeiten und Löschen von Associations (zwischen Topics)
- Erstellen, Bearbeiten und Löschen von Occurrences und Wissensatomen

7.1.2.3 Klasse OntologyManager

`OntologyManager` bietet Methoden zur **Verwaltung des Bestands an Topic Maps**. Mit Hilfe von `OntologyManager` können Topic Maps als Ganzes neu erstellt, bearbeitet oder gelöscht werden. Der Zugriff auf die Topic Maps ist, wie bei `Navigator` und `TMEditor`, über eine **grafische Benutzungsschnittstelle** möglich. Anders als bei den beiden erst genannten Klassen ist die Darstellung der Benutzungsschnittstelle nicht so eingeschränkt, da lediglich **Metadaten** einer Topic Map darzustellen sind. Diese Daten können grafisch wesentlich weniger aufwändig dargestellt werden.

OntologyManager stellt folgende Features zum Lesen und Schreiben von Eigenschaften (Metadaten) einer Topic Map zur Verfügung (vgl. Abbildung 107):

- Erstellen einer neuen, leeren Topic Map
- Bearbeiten der Eigenschaften einer Topic Map
- Löschen einer Topic Map (inclusive aller seiner Objekte)
- Importieren einer Topic Map, die als Datei im xtm-Format (vgl. [XTM dtd, 2001]) lokal vorhanden ist, als neue Topic Map in die Contentpool-Verwaltung
- Merging, also Verschmelzen, von Topic Maps

Zu beachten gilt es, dass einzelne Objekte in Topic Maps (z.B. Topics, Associations, usw.) mit Hilfe von OntologyManager nicht bearbeitet werden können. Zu diesem Zweck müssen die Funktionen der Klasse TMEditor verwendet werden.

7.1.2.4 Klasse SearchTool

SearchTool bietet Algorithmen zum **Suchen innerhalb der Objekte** einer Topic Map an. Da die Such-Algorithmen komplex sind, ist in SearchTool lediglich die **Anwendungslogik** für die Suche implementiert. SearchTool bietet von sich aus keine grafische Benutzungsschnittstelle an. Es wird lediglich eine Schnittstelle zum Zugriff auf die Such-Algorithmen zur Verfügung gestellt.

SearchTool stellt folgende Funktionalität zur Suche in den Objekten einer Topic Map zur Verfügung (vgl. Abbildung 107):

- **Einfache Stichwortsuche** in den Topics einer Topic Map, wobei die Basenames aller Topics nach dem Suchbegriff durchsucht werden
- **Fortgeschrittene Suche**, wobei zusätzlich zu einem Suchbegriff semantische Aspekte als Parameter berücksichtigt werden können (wie z.B. „ist an Association vom Typ ‚liegt_in' beteiligt", „ist vom Topic Type ‚Stadt'", usw.)

Das Ziel bei der Anwendung der genannten Suchalgorithmen besteht stets darin, Topics zu finden, die auf die spezifizierten Suchmuster passen. Nach Objekten wie *Associations* oder *Occurrences* kann in der aktuellen Version von SearchTool lediglich indirekt gesucht werden. Dies wird möglich, indem eines dieser Objekte als Parameter für die fortgeschrittene Suche spezifiziert wird.

7.1.2.5 Klasse Topicsearch

Topicsearch implementiert die **grafische Benutzungsschnittstelle**, über welche die **Such-Algorithmen** in der Klasse SearchTool verwendet werden können. Daraus ergeben sich auch die von Topicsearch angebotenen Features (vgl. Abbildung 107):

- Zugriff auf die einfache Stichwortsuche von SearchTool
- Zugriff auf die fortgeschrittene Suche von SearchTool

7.1.2.6 Sub-Package „backend“

Das Sub-Package `backend` im Package `scholion.contentpool` bildet das „Rückgrat“ der Contentpool-Verwaltung zu einer **relationalen Datenbank**. Es dient also dazu, die Daten der Contentpool-Verwaltung aus einer relationalen Datenbank zu lesen oder dieselben dorthin zu schreiben. Das Sub-Package ist folglich für die Herstellung und Erhaltung eines **konsistenten Zustands** der Daten in der Contentpool-Verwaltung verantwortlich.

Folgende Klassen sind im Package `scholion.contentpool.backend` enthalten:

- Klasse TopicMapProcessorDB (Punkt 7.1.2.6.1)
- Klasse HibernateTopicMapProcessor (Punkt 7.1.2.6.2)
- Klasse BackupThread (Punkt 7.1.2.6.3)

Abbildung 108 zeigt das Klassenmodell der oben genannten Klassen.

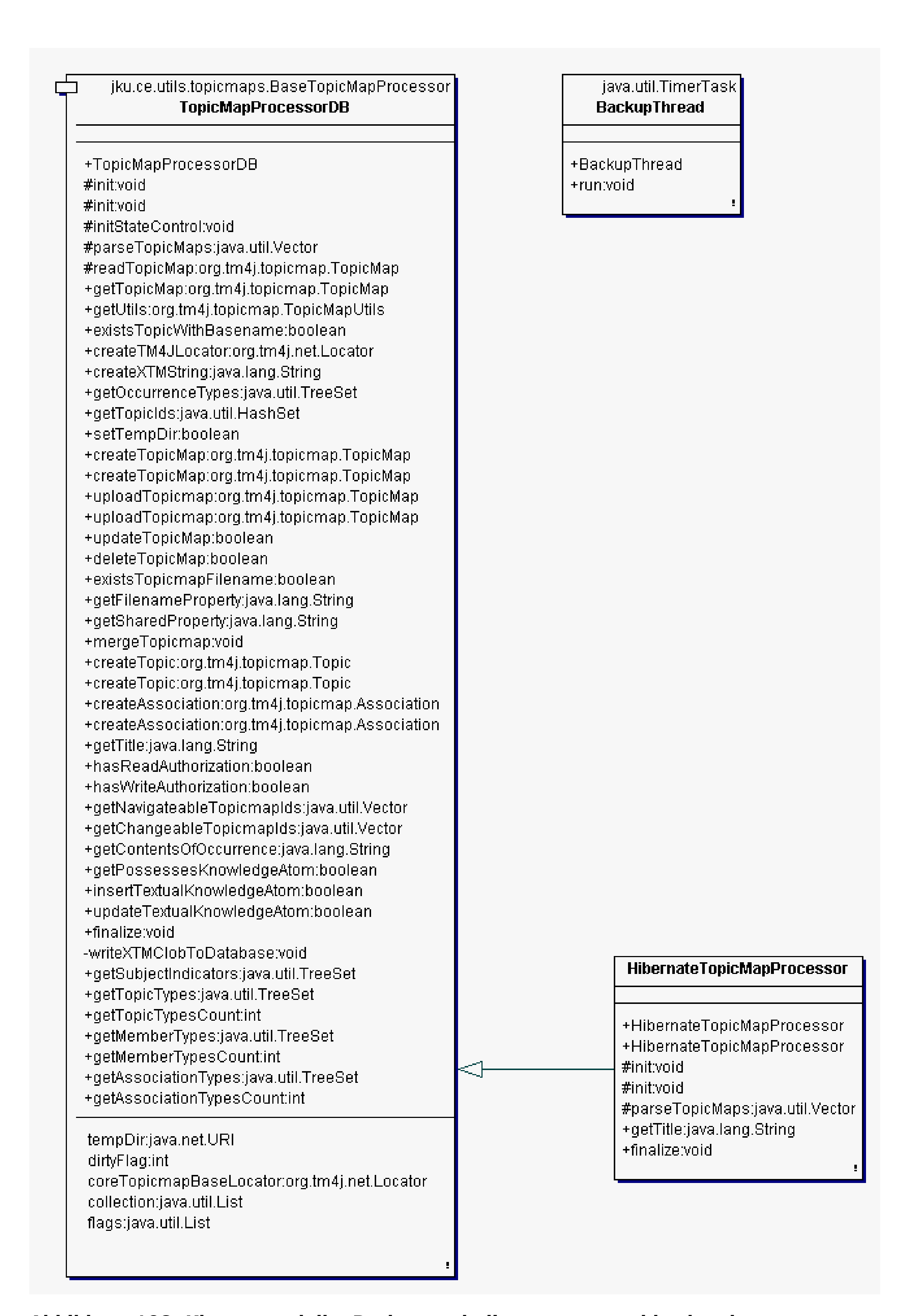


Abbildung 108: Klassenmodell – Package scholion.contentpool.backend

7.1.2.6.1 Klasse TopicMapProcessorDB

`TopicMapProcessorDB` ist die Instanz, welche für die **Übermittlung von Daten** zwischen einer relationalen Datenbank und der Contentpool-Verwaltung in Scholion WB+ zuständig ist. Es werden **grundlegende Dienste** angeboten, die von fast allen Klassen der Contentpool-Verwaltung benötigt werden, um Operationen auf Topic Maps durchzuführen. `TopicMapProcessorDB` verwaltet alle im System vorhandenen Topic Maps im Speicher (in Java-Objekten) und sorgt dafür, dass diese Objekte regelmäßig in der Datenbank als persistenten Speicher geschrieben werden.

`TopicMapProcessorDB` ist eine Erweiterung der Klasse `BaseTopicMapProcessor` (vgl. Abschnitt 7.1.4.1, Seite 266). Folgende Dienste werden von `TopicMapProcessorDB` für die Klassen der Contentpool-Verwaltung angeboten (vgl. Abbildung 108):

- **Parsen** von Topic Maps, d.h. das Erstellen einer Struktur von Java-Objekten, die in der Lage ist, eine Topic Map im xtm-Format (vgl. [XTM dtd, 2001]) äquivalent als Objekt-Sammlung darzustellen
- Erstellen einer Zeichenkette im **xtm-Format** (vgl. [XTM dtd, 2001]), welche anschließend in einer Datei gespeichert werden kann
- **Erstellen** einer neuen leeren Topic Map
- **Verändern** der Metadaten einer Topic Map
- **Löschen** einer bestehenden Topic Map
- Zugriff auf die **Objekte** von bestehenden Topic Maps
- Erstellen, Bearbeiten und Löschen von Objekten in bestehenden Topic Maps
- **Statusverwaltung** für Topic Maps (mögliche Zustände: verändert / unverändert)
- Verwaltung von **Zugriffsrechten** von Benutzern der Contentpool-Verwaltung auf bestehende Topic Maps

Bei der Implementierung der genannten Dienste macht `TopicMapProcessorDB` intensiven Gebrauch von der **Klassenbibliothek tm4j** (vgl. [TM4J, 2002]).

7.1.2.6.2 Klasse HibernateTopicMapProcessor

Wie aus dem Klassenmodell in Abbildung 108 ersichtlich wird, bietet `HibernateTopicMapProcessor` nahezu die selben Dienste an wie seine Superklasse `TopicMapProcessorDB`. Der Unterschied zwischen den beiden Klassen besteht in der Art und Weise, wie Anbindung an eine relationale Datenbank implementiert ist.

`HibernateTopicMapProcessor` verwendet dabei das **Werkzeug „Hibernate"** (vgl. [Hibernate, 2003]). Hibernate ist ein Werkzeug, mit dessen Hilfe Java-Objekte schema-transparent auf eine relationale Datenbank abgebildet werden können. „**Schematransparenz**" bedeutet, dass Hibernate aus den gegebenen Java-Objekten selbständig ein relationales Datenbank-Schema entwickelt. Der Zugriff auf die Methoden des Hibernate Werkzeugs erfolgt auf sehr hoher Abstraktionsebene (Java-Objekte werden direkt in die Schnittstelle übergeben und von dort direkt wieder gelesen) [Hibernate, 2003].

Das viel versprechende Konzept des Hibernate Werkzeugs hat sich nach umfangreichen Tests als **weniger performant** erwiesen als jenes, das in `TopicMapProcessorDB` verfolgt wurde. Aus diesem Grund wurde `HibernateTopicMapProcessor` lediglich in einem experimentellen Stadium implementiert. Innerhalb der Contentpool-Verwaltung besitzt `HibernateTopicMapProcessor` keine praktische Bedeutung.

7.1.2.6.3 Klasse BackupThread

`BackupThread` wird von TopicMapProcessorDB verwendet, um das Intervall und die Zeitpunkte für die **Synchronisation der Daten** in der relationalen Datenbank mit den Java-Objekten im Arbeitsspeicher zu steuern.

Folgende Funktionalität wird von `BackupThread` angeboten:

- Konfiguration des Intervalls für das Schreiben der Daten aus den Topic Maps in die relationale Datenbank
- Durchführen des Schreibevorgangs auf die relationale Datenbank

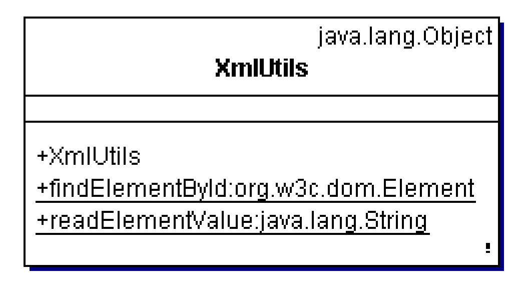
Aus Gründen der Effizienz nützt `BackupThread` die **Statusverwaltung** von `TopicMapProcessorDB` und überträgt nur die Daten von jenen Topic Maps, deren Status auf „verändert" gesetzt ist, in die relationale Datenbank. Die Daten aller anderen Topic Maps werden nicht in die relationale Datenbank geschrieben.

7.1.3 Package „scholion.util"

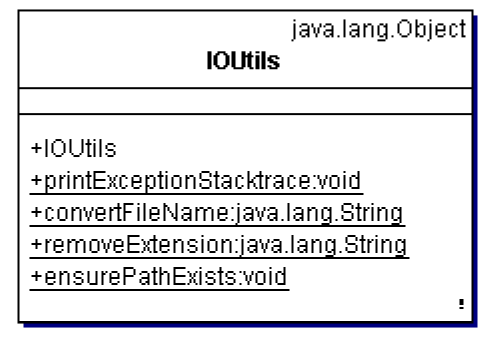
Das Package `scholion.util` enthält Klassen und **Werkzeug-Klassen**, deren Funktionalität sowohl von der Contentpool-Verwaltung als auch von anderen Teilen der Wissenstransfer-Umgebung Scholion WB+ in Anspruch genommen wird. Es besteht aus den folgenden, in den in Klammern genannten Abschnitten beschriebenen Teilen:

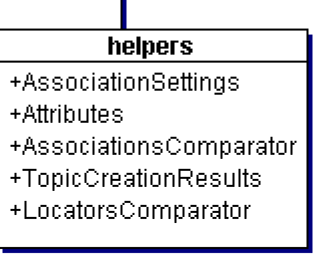
- Klasse XmlUtils (Abschnitt 7.1.3.1)
- Klasse IOUtils (Abschnitt 7.1.3.2)
- Klasse FilteringTopicMapPrettyPrinter (Abschnitt 7.1.3.3)
- Klasse TopicMapUtils (Abschnitt 7.1.3.4)
- Sub-Package „helpers" (Abschnitt 7.1.3.5)

Abbildung 109 gibt einen Überblick über die oben genannten Teile des Packages `scholion.util`.









java.lang.Object
TopicMapUtils
+TopicMapUtils
+getTopicTypesXML:java.lang.String
+getTopicListXML:java.lang.String
+getTopicTreeXML:java.lang.String
+getEmptyTypesTreeSection:java.lang.String
+getEmptyTreeSection:java.lang.String
+getAssociationTypesXML:java.lang.String
+getMemberTypesXML:java.lang.String
+getSubjectsXML:java.lang.String
+getScopesXML:java.lang.String
+getOntologySubmenu:java.lang.String
+filterOccurrenceData:java.lang.String

Abbildung 109: Klassenmodell – Package scholion.util

7.1.3.1 Klasse XmlUtils

`XmlUtils` bietet Funktionalität für die **Behandlung von Daten im XML-Format** an. Da Topic Maps im xtm-Format, welches auf XML basiert (vgl. [XTM dtd, 2001]), gespeichert werden können, ist eine umfangreiche Funktionalität für den Umgang mit XML von Nöten. Ansonsten könnten Topic Maps weder importiert noch exportiert werden.

Folgende Funktionalität wird von `XmlUtils` implementiert (vgl. Abbildung 108):

- Extrahieren eines Wertes eines XML-Elements als Zeichenkette (Umwandlung eines #CDATA-Elements in einen Java-String)
- Finden eines XML-Elements mit einer bestimmten Zeichenkette zur eindeutigen Identifizierung innerhalb des Element-Baums eines XML-Elements[35]

7.1.3.2 Klasse IOUtils

`IOUtils` bietet allgemeine Dienste im Zusammenhang mit Lesen oder Schreiben von Daten (**I**nput / **O**utput) in oder aus **Datenströmen** an.

Folgende Dienste werden von `IOUtils` unterstützt (vgl. Abbildung 108):

- Ausgabe detaillierter Information zu Fehlersituationen an den Benutzer
- Umwandlung von Dateinamen in Bezeichnungen für Topic Maps
- Anlegen von verschachtelten Verzeichnisstrukturen im lokalen Dateisystem

7.1.3.3 Klasse FilteringTopicMapPrettyPrinter

`FilteringTopicMapPrettyPrinter` ist eine Utility-Klasse zur **Erzeugung von Zeichenketten** aus *BaseNames* eines Topics. Bei der Erzeugung der Zeichenketten werden Ersetzungen von bestimmten, für die Darstellung eines Topics in XML oder HTML problematischen Zeichen vorgenommen. Somit erzeugt `FilteringTopicMapPrettyPrinter` eine textuelle Darstellung eines Topics, die besonders gut für die Einbindung in XML- oder HTML-Code geeignet ist.

Folgende Funktionalität wird von `FilteringTopicMapPrettyPrinter` angeboten (vgl. Abbildung 108; siehe dazu auch die Beschreibung der Superklasse `TopicMapPrettyPrinter`, Abschnitt 7.1.4.3, Seite 267):

- Umwandlung eines Topic-*BaseNames* in eine Zeichenkette, die als textuelle Beschreibung eines Topics dient
- Umwandlung aller *BaseNames* eines Topics in eine zusammen gesetzte Zeichenkette
- Unterdrückung (Filterung) aller für die Darstellung in XML oder HTML problematischen Zeichen (z.B. Anführungszeichen)

7.1.3.4 Klasse TopicMapUtils

`TopicMapUtils` ist eine Utility-Klasse, deren Zweck darin besteht, **Information über eine Topic Map** in XML-Code einzuarbeiten, der anschließend für die **Generierung einer Benutzungsschnittstelle** herangezogen wird. Information über die Topic Map er-

---

[35] Hinweis: in der Implementierung des Document Object Model (DOM) für Java befindet sich offensichtlich ein Fehler, der zu einem nicht der Spezifikation entsprechenden Verhalten der Methode `getElementById` führt. `XmlUtils` schließt diese offenbar bestehende Lücke.

hält TopicMapUtils aus Objekten, die an der Schnittstelle durch den TopicMapProcessorDB als Eingabeparameter übergeben werden.

Folgende Funktionalität wird von TopicMapUtils angeboten (vgl. Abbildung 108):

- Erzeugung von XML Code zur Beschreibung von Information über Topic Types, Association Types, Occurrence Types, Scopes, subject identifiers, Member Types und so weiter
- Unterdrückung (Filterung) aller für die Darstellung in XML oder HTML problematischen Zeichen (z.B. Anführungszeichen)

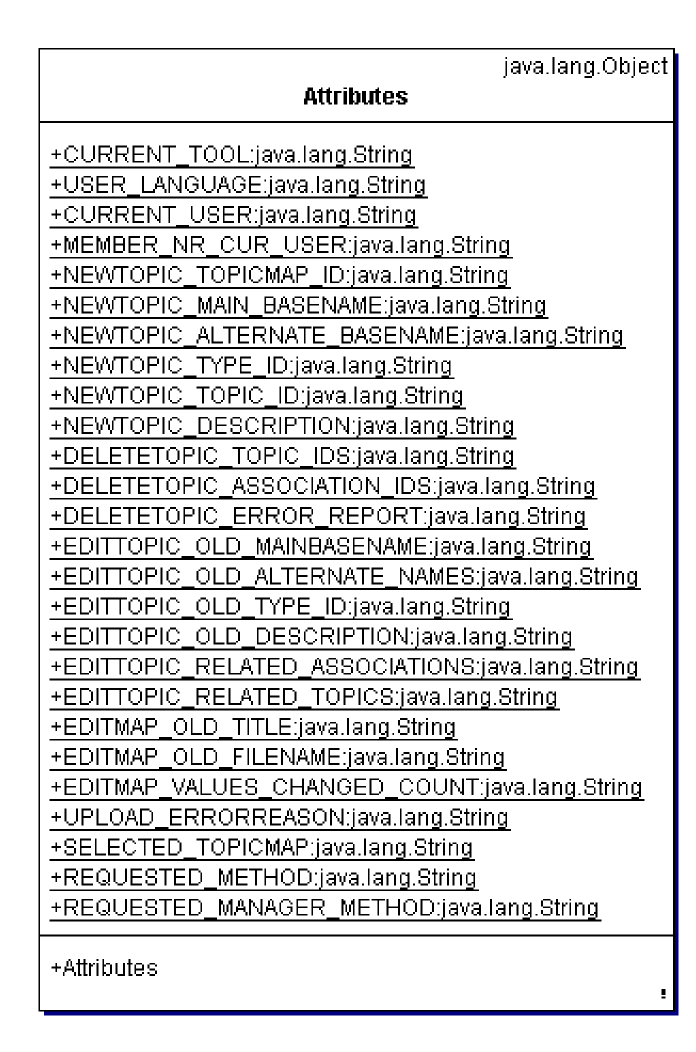
7.1.3.5 Sub-Package „helpers“

Das Sub-Package helpers im Package scholion.util enthält nützliche Klassen, die für die Verwendung durch alle Teile der Wissenstransfer-Umgebung Scholion WB+ vorgesehen sind. Die Funktionalität im Sub-Package helpers hat nur teilweise mit der Contentpool-Verwaltung (bzw. mit Topic Maps) zu tun.

Folgende Klassen sind im Package scholion.util.helpers enthalten:

- Klasse Attributes (Punkt 7.1.3.5.1)
- Klasse AssociationSettings (Punkt 7.1.3.5.2)
- Klasse TopicCreationResults (Punkt 7.1.3.5.3)
- Klasse AssociationsComparator (Punkt 7.1.3.5.4)
- Klasse LocatorsComparator (Punkt 7.1.3.5.5)

Abbildung 110 zeigt das Klassenmodell der oben genannten Klassen.



java.lang.Object
java.util.Comparator
java.io.Serializable
AssociationsComparator
+AssociationsComparator
+AssociationsComparator
+compare:int
+equals:boolean

java.lang.Object
java.util.Comparator
java.io.Serializable
LocatorsComparator
+LocatorsComparator
+compare:int
+equals:boolean

java.lang.Object
AssociationSettings
+AssociationSettings
+AssociationSettings
+AssociationSettings
+AssociationSettings
+setPlayer:void
id:java.lang.String
typeDefId:java.lang.String
typeDefName:java.lang.String
players:java.util.Map

java.lang.Object
TopicCreationResults
+TopicCreationResults
+wasSuccess:boolean
topic:org.tm4j.topicmap.Topic
message:java.lang.String

Abbildung 110: Klassenmodell – Package scholion.util.helpers

7.1.3.5.1 Klasse Attributes

`Attributes` enthält eine Sammlung von statischen, öffentlich sichtbaren **Zeichenketten** zur Beschreibung der **Namen von Parametern**, die zur Speicherung eines bestimmten Status (z.B. der vom Benutzer getroffenen Entscheidungen bei der Bearbeitung der Associations eines Topics) einer Sitzung im **Session-Objekt eines Servlets** (vgl. [Sun, 1999], [Java, 2002]) gesetzt werden. Die Verwendung der `Attributes`-Klasse erleichtert das Setzen und Lesen der Sitzungs-Parameter.

Funktionalität (d.h. Methoden) ist in der Klasse `Attributes` keine implementiert (vgl. Abbildung 110).

7.1.3.5.2 Klasse AssociationSettings

`AssociationSettings` dient zur **Modellierung von Information** über eine *Association* zwischen zwei oder mehr Topics. Instanzen der Klasse `AssociationSettings` werden vor allem von `TMEditor` (vgl. Abschnitt 7.1.2.2, Seite 253) erzeugt und verwendet, um die vom Benutzer spezifizierten Daten über gewünschte *Associations* in einem Java-Objekt zwischenzuspeichern.

`AssociationSettings` bietet folgende Funktionalität an (vgl. Abbildung 110):

- Erzeugung einer neuen Instanz zur Modellierung von Information über eine *Association* mit Hilfe mehrerer (überladener) Konstruktoren
- Zugriff auf die Information über eine Association (eindeutige Id, Typ der *Association*, beteiligte Topics) mit Hilfe entsprechender `get`-Methoden

7.1.3.5.3 Klasse TopicCreationResults

`TopicCreationResults` dient zur **Modellierung der Ergebnisse** eines Versuchs zur Schaffung eines neuen Topics. Dabei können zwei Möglichkeiten unterschieden werden:

1. Das Topic wurde erfolgreich angelegt. In diesem Fall enthält die Instanz von `TopicCreationResults` eine positive Markierung (mit Hilfe eines booleschen Wertes) sowie das neu geschaffene Topic-Objekt.
2. Das Topic konnte nicht angelegt werden. In diesem Fall enthält die Instanz von `TopicCreationResults` eine negative Markierung (mit Hilfe eines booleschen Wertes) sowie eine Fehlermeldung, die Information über den fehlerhaften Zustand beim Versuch der Erzeugung des neuen Topics enthält. Weitere Objekte sind im Fall eines Miss-Erfolgs nicht verfügbar.

Zur Realisierung des oben beschriebenen Verhaltens bietet `TopicCreationResults` folgende Funktionalität an (vgl. Abbildung 110):

- Erzeugen einer neuen Instanz (dies wird automatisch von der Klasse `TMEditor` vorgenommen)
- Zugriff auf die Objekte von TopicCreationResults (Erfolgs-Status, neu erzeugtes Topic im Fall eines Erfolgs, Fehlermeldung im Fall eines Misserfolgs) mit Hilfe entsprechender get-Methoden

7.1.3.5.4 Klasse AssociationsComparator

`AssociationsComparator` erfüllt das Java-Interface `java.util.Comparator`. Es dient dazu, eine **Ordnungsrelation für Objekte** herzustellen, die das Interface *Association* erfüllen. Damit wird es möglich, *Associations* mit Hilfe der in `AssociationsComparator` definierten formalen Regeln zu ordnen. Die formale Ordnung beeinflusst die Reihenfolge, in der eine Menge von *Associations* an der Benutzungsschnittstelle präsentiert wird.

Die Funktionalität der Klasse `AssociationsComparator` ist durch das Interface `java.util.Comparator` hinreichend definiert (vgl. Abbildung 110).

7.1.3.5.5 Klasse LocatorsComparator

`LocatorsComparator` erfüllt das Java-Interface `java.util.Comparator`. Es dient dazu, eine **Ordnungsrelation für Objekte** herzustellen, die Instanzen der Klasse *Locator* sind. Damit wird es möglich, *Locators* mit Hilfe der in `LocatorsComparator` definierten formalen Regeln zu ordnen. Die formale Ordnung beeinflusst die Reihenfolge, in der eine Menge von *Locators* an der Benutzungsschnittstelle präsentiert wird.

Die Funktionalität der Klasse `LocatorsComparator` ist durch das Interface `java.util.Comparator` hinreichend definiert (vgl. Abbildung 110).

7.1.4 Package „utils.topicmaps“

Das Package `utils.topicmaps` verfolgt den Zweck, Algorithmen für die **Verwaltung von Topic Maps** auf einem hohen Abstraktionsniveau anzubieten. Mit seiner Hilfe wird die komfortable Manipulation von Topic Maps ermöglicht, wobei die physische Speicherung derselben transparent bleibt.

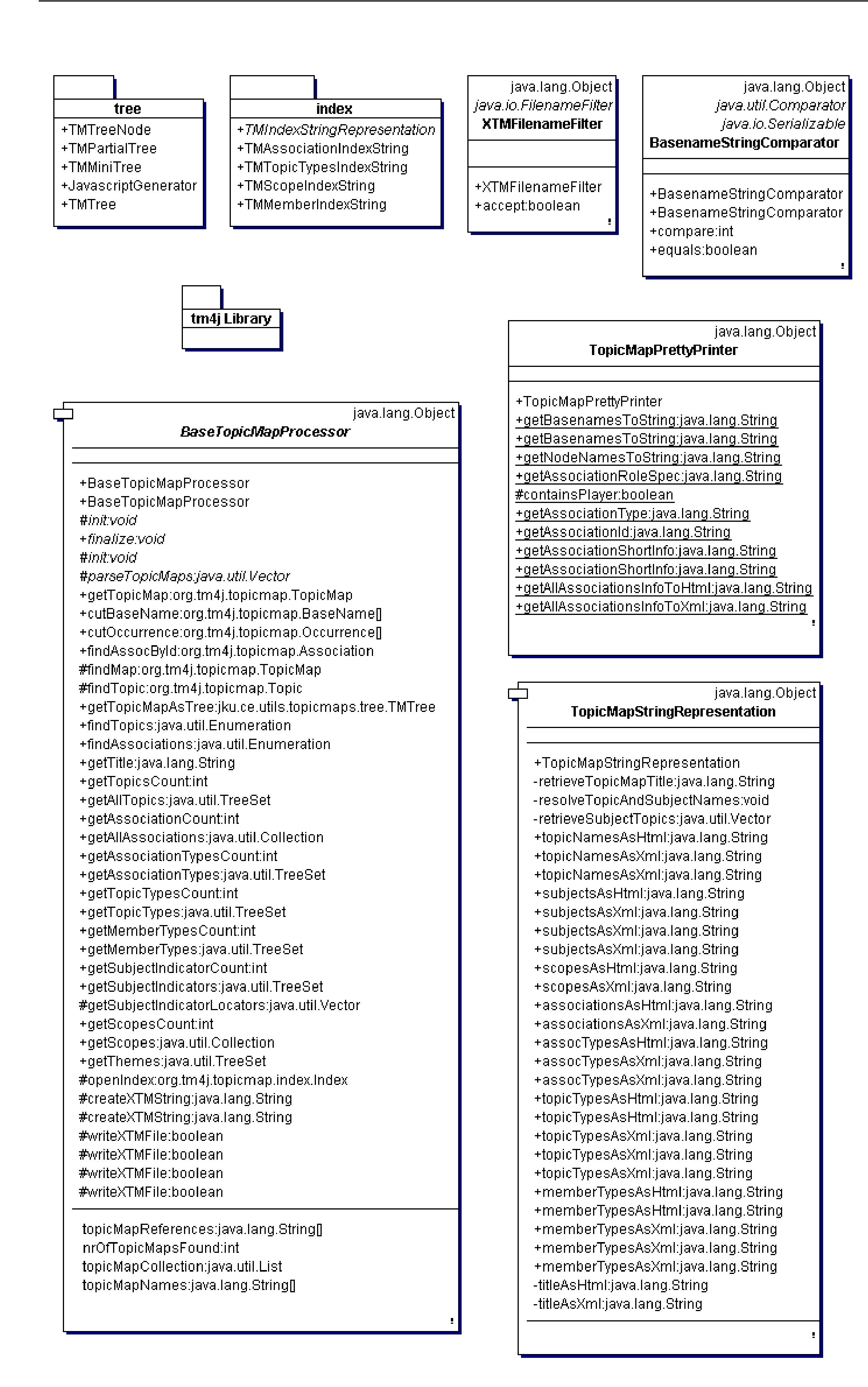


Abbildung 111: Klassenmodell – Package utils.topicmaps

Das Package `utils.topicmaps` besteht aus den folgenden, in den in Klammern genannten Abschnitten beschriebenen Teilen:

- Klasse BaseTopicMapProcessor (Abschnitt 7.1.4.1)
- Klasse TopicMapStringRepresentation (Abschnitt 7.1.4.2)
- Klasse TopicMapPrettyPrinter (Abschnitt 7.1.4.3)
- Klasse BasenameStringComparator (Abschnitt 7.1.4.4)
- Klasse XTMFilenameFilter (Abschnitt 7.1.4.5)
- Sub-Package „index" (Abschnitt 7.1.4.6)
- Sub-Package „tree" (Abschnitt 7.1.4.7)

Abbildung 111 (siehe oben) gibt einen Überblick über die oben genannten Teile des Packages `utils.topicmaps`.

7.1.4.1 Klasse BaseTopicMapProcessor

`BaseTopicMapProcessor` ist die **zentrale Instanz** für die Verwaltung von Topic Maps, welche für die **Manipulation** und die **persistente Speicherung** der vom Benutzer erstellten Topic Maps zuständig ist. Es werden grundlegende Dienste angeboten, die benötigt werden, um Operationen auf Topic Maps durchzuführen. `BaseTopicMapProcessor` verwaltet alle im System vorhandenen Topic Maps im Speicher (in Java-Objekten) und sorgt dafür, dass diese Objekte regelmäßig in einen persistenten Speicher (z.B. eine relationale Datenbank) geschrieben werden.

Folgende Dienste werden von `BaseTopicMapProcessor` angeboten (vgl. Abbildung 111):

- **Parsen** von Topic Maps, d.h. das Erstellen einer Struktur von Java-Objekten, die in der Lage ist, eine Topic Map im xtm-Format (vgl. [XTM dtd, 2001]) äquivalent als Objekt-Sammlung darzustellen
- Erstellen einer Zeichenkette im **xtm-Format** (vgl. [XTM dtd, 2001]), welche anschließend in einer Datei gespeichert werden kann
- **Erstellen** einer neuen leeren Topic Map
- **Verändern** der Metadaten einer Topic Map
- **Löschen** einer bestehenden Topic Map
- Zugriff auf die **Objekte** von bestehenden Topic Maps
- Erstellen, Bearbeiten und Löschen von Objekten in bestehenden Topic Maps
- **Statusverwaltung** für Topic Maps (mögliche Zustände: verändert / unverändert)

Bei der Implementierung der genannten Dienste macht `BaseTopicMapProcessor` intensiven Gebrauch von der **Klassenbibliothek tm4j** (vgl. [TM4J, 2002]).

7.1.4.2 Klasse TopicMapStringRepresentation

`TopicMapStringRepresentation` dient zur Erzeugung einer stark vereinfachten, **textuellen Darstellung** der gesamten Inhalte aus einer Topic Map. Die textuelle Darstellung wird als einfache Zeichenkette gespeichert und kann somit entweder an einer grafischen Benutzungsschnittstelle präsentiert, für die Weitergabe an andere Personen gespeichert oder für Präsentationszwecke ausgedruckt werden. `TopicMapStringRepresentation` ist **einfach lesbar**, jedoch **weniger mächtig** als die formale Beschreibungssprache xtm (vgl. [XTM dtd, 2001], [TM Strict, 2000]).

Eine Instanz von `TopicMapStringRepresentation` wird aus einem `TopicMap` Objekt erzeugt. Folgende Funktionalität wird angeboten (vgl. Abbildung 111):

- Zugriff auf die Metadaten einer Topic Map in textueller Form
- Zugriff auf die textuelle Darstellung einzelner Objekte (z.B. Topics, Associations, Occurrences) aus einer Topic Map

7.1.4.3 Klasse TopicMapPrettyPrinter

`TopicMapPrettyPrinter` ist eine Utility-Klasse zur **Erzeugung von Zeichenketten** aus *BaseNames* eines Topics. Somit erzeugt `TopicMapPrettyPrinter` eine textuelle Darstellung eines Topics, die für die rein textuelle Darstellung einzelner Topics aus Topic Maps geeignet ist.

Folgende Funktionalität wird von `TopicMapPrettyPrinter` angeboten (vgl. Abbildung 111):

- Umwandlung eines Topic-*BaseNames* in eine Zeichenkette, die als textuelle Beschreibung eines Topics dient
- Umwandlung aller *BaseNames* eines Topics in eine zusammen gesetzte Zeichenkette

7.1.4.4 Klasse BasenameStringComparator

`BasenameStringComparator` erfüllt das Java-Interface `java.util.Comparator`. Es dient dazu, eine **Ordnungsrelation für Objekte** herzustellen, die das Interface *Basename* erfüllen. Damit wird es möglich, *Basenames* eines Topics oder anderer `Topicmap`-Objekte mit Hilfe der in `BasenameStringComparator` definierten formalen Regeln zu ordnen. Die formale Ordnung beeinflusst die Reihenfolge, in der eine Menge von *Basenames* an der Benutzungsschnittstelle präsentiert wird.

Die Funktionalität der Klasse `BasenameStringComparator` ist durch das Interface `java.util.Comparator` hinreichend definiert (vgl. Abbildung 111).

7.1.4.5 Klasse XTMFilenameFilter

`XTMFilenameFilter` erfüllt das Java-Interface `java.io.FilenameFilter`. Es dient dazu, im **lokalen Dateisystem** eines Benutzers Dateien zu erkennen, die der Spezifikation des **xtm-Formats** (vgl. [XTM dtd, 2001]) gehorchen. Somit können mit Hilfe von `XTMFile-`

nameFilter Dateien erkannt werden, die potenzielle Kandidaten für den Import einer Topic Map in den BaseTopicMapProcessor sind.

Die Funktionalität der Klasse XTMFilenameFilter ist durch das Interface java.io.FilenameFilter hinreichend definiert (vgl. Abbildung 111).

7.1.4.6 Sub-Package „index“

Das Sub-Package index im Package utils.topicmaps dient zur textuellen Repräsentation von **Index-Strukturen**. Ein Index ist eine Datenstruktur, die einer bestimmten Sicht auf eine Topic Map entspricht und z.B. alle Topic Types einer Topic Map enthält (vgl. [TM4J, 2002]).

Folgende Klassen und Interfaces sind im Package utils.topicmaps.index enthalten:

- Interface TMIndexStringRepresentation (Punkt 7.1.4.6.1)
- Klasse TMAssociationIndexString (Punkt 7.1.4.6.2)
- Klasse TMMemberIndexString (Punkt 7.1.4.6.3)
- Klasse TMScopeIndexString (Punkt 7.1.4.6.4)
- Klasse TMTopicTypesIndexString (Punkt 7.1.4.6.5)

Abbildung 112 zeigt das Klassenmodell der oben angeführten Klassen und Interfaces.

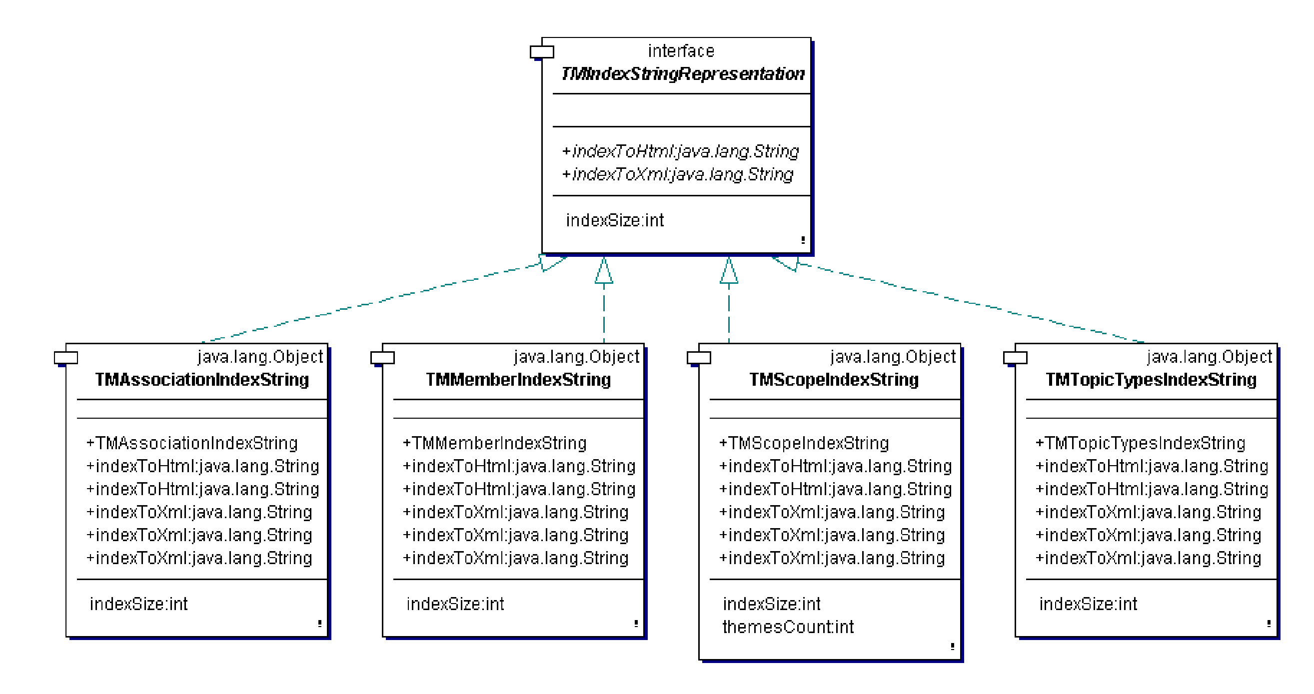


Abbildung 112: Klassenmodell – Package utils.topicmaps.index

7.1.4.6.1 Interface TMIndexStringRepresentation

Das Interface TMIndexStringRepresentation definiert die Schnittstelle für Klassen zur **textuellen Repräsentation von Indexstrukturen** auf die Objekte in einer Topic Map. Es enthält die beiden folgenden Methoden (vgl. Abbildung 112):

- Die Methode `indexToHtml()` berechnet eine Darstellung des Index in HTML-Code.
- Die Methode `indexToXml()` berechnet eine Darstellung des Index in XML-Code.

7.1.4.6.2 Klasse TMAssociationIndexString

`TMAssociationIndexString` implementiert das Interface `TMIndexStringRepresentation` in Bezug auf die *Association Types* in einer Topic Map. Eine Instanz von `TMAssociationIndexString` wird aus einem `TopicMap`-Objekt erzeugt und berechnet daraus alle für eine angemessene Darstellung der *Association Types* benötigte Information. Es bietet damit Funktionalität, um den *Association Types Index* einer Topic Map entweder als HTML oder als XML darzustellen (vgl. Abbildung 112).

7.1.4.6.3 Klasse TMMemberIndexString

`TMMemberIndexString` implementiert das Interface `TMIndexStringRepresentation` in Bezug auf die *Member Types* in einer Topic Map. Eine Instanz von `TMMemberIndexString` wird aus einem `TopicMap`-Objekt erzeugt und berechnet daraus alle für eine angemessene Darstellung der *Member Types* benötigte Information. Es bietet damit Funktionalität, um den *Member Types Index* einer Topic Map entweder als HTML oder als XML darzustellen (vgl. Abbildung 112).

7.1.4.6.4 Klasse TMScopeIndexString

`TMScopeIndexString` implementiert das Interface `TMIndexStringRepresentation` in Bezug auf die *Scopes* (vgl. dazu die theoretischen Erläuterungen in den beiden Abschnitten 4.4.2.2.6 Namenskonventionen innerhalb eines Gültigkeitsbereichs, Seite 84 ff., und 4.4.2.3.1 Themen („Topics") und Gegenstände („Subjects"), Seite 86 ff.) in einer Topic Map. Eine Instanz von `TMScopesIndexString` wird aus einem `TopicMap`-Objekt erzeugt und berechnet daraus alle für eine angemessene Darstellung der *Scopes* und *Subjects* benötigte Information. Es bietet damit Funktionalität, um den *Scope Index* einer Topic Map entweder als HTML oder als XML darzustellen (vgl. Abbildung 112).

7.1.4.6.5 Klasse TMTopicTypesIndexString

`TMTopicTypesIndexString` implementiert das Interface `TMIndexStringRepresentation` in Bezug auf die *Topic Types* in einer Topic Map. Eine Instanz von `TMTopicTypesIndexString` wird aus einem `TopicMap`-Objekt erzeugt und berechnet daraus alle für eine angemessene Darstellung der *Topic Types* benötigte Information. Es bietet damit Funktionalität, um den *Topic Types Index* einer Topic Map entweder als HTML oder als XML darzustellen (vgl. Abbildung 112).

7.1.4.7 Sub-Package „tree"

Das Sub-Package `tree` im Package `utils.topicmaps` dient zur Berechnung der Darstellung einer Topic Map in Form eines Baumes. Mit der Darstellung allein ist es jedoch nicht getan; Zweck der Baumdarstellung ist es auch, dem Benutzer das Navigieren durch die Topics einer Topic Map auf einfache und verständliche Weise zu ermöglichen. Realisiert wird die Baumdarstellung mit Hilfe von **JavaScript**.

Folgende Klassen sind im Package `utils.topicmaps.tree` enthalten:

- Klasse JavascriptGenerator (Punkt 7.1.4.7.1)
- Klasse TMTreeNode (Punkt 7.1.4.7.2)
- Klasse TMTree (Punkt 7.1.4.7.3)
- Klasse TMPartialTree (Punkt 7.1.4.7.4)
- Klasse TMMiniTree (Punkt 7.1.4.7.5)

Abbildung 113 zeigt das Klassenmodell der oben angeführten Klassen.

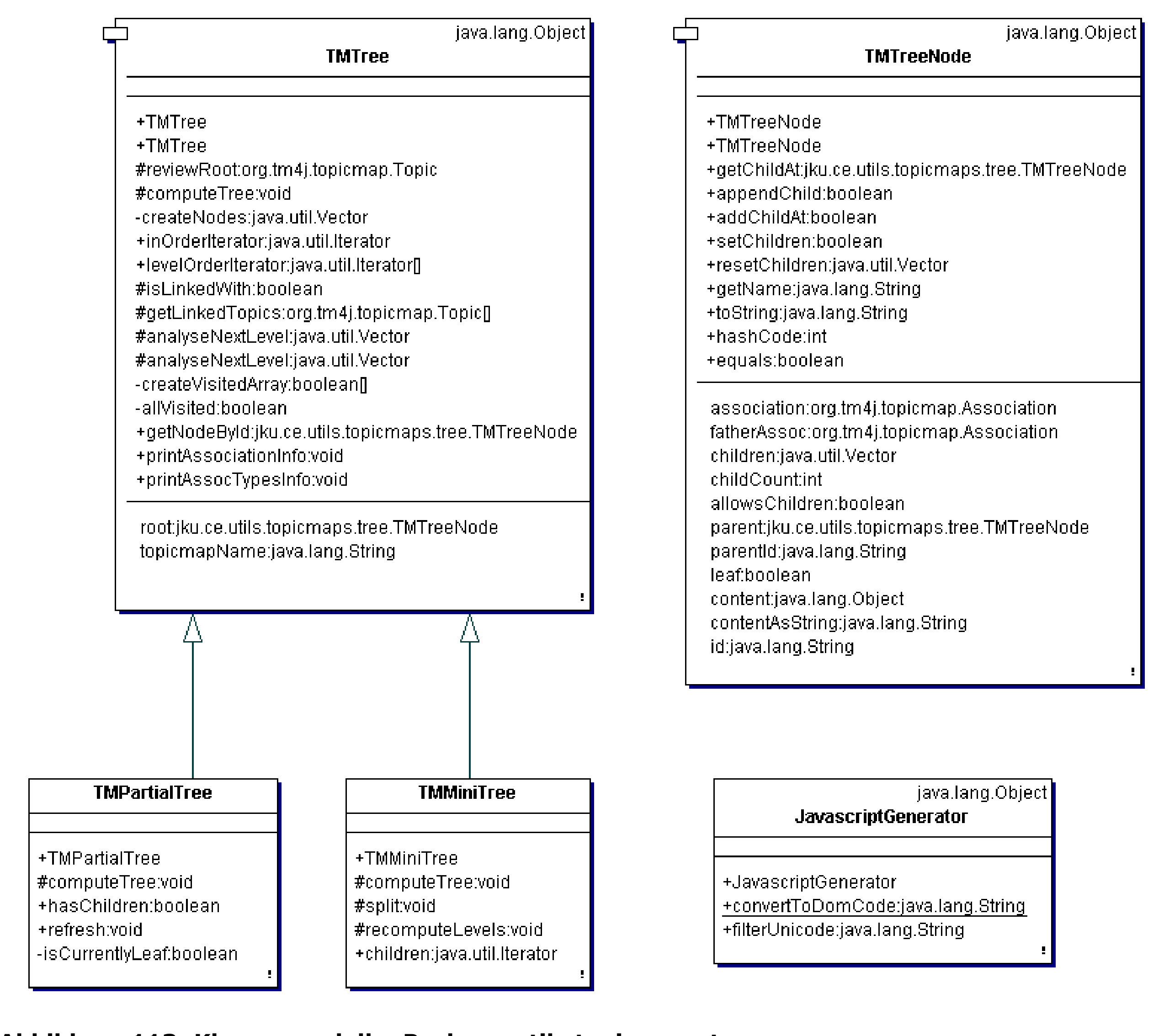


Abbildung 113: Klassenmodell – Package utils.topicmaps.tree

7.1.4.7.1 Klasse JavascriptGenerator

Die Klasse `JavascriptGenerator` bildet die **Schnittstelle** zwischen dem in diesem Abschnitt erläuterten Package `utils.topicmaps.tree` (d.h. der Baumdarstellung einer Topic Map in Java) und dem JavaScript „`tm_navigate`" (vgl. Abbildung 47: Erweitertes

XML-spezifisches MVC-Konzept von Scholion WB+, rechter Teil), das zur Darstellung auf dem Rechner des Benutzers benötigt wird. Aufgabe von `JavascriptGenerator` ist es, eine gegebene Topic Map so in **JavaScript-Code** umzusetzen, dass dieser Code von „`tm_navigate`" verstanden wird und das Script eine **authentische Darstellung** der Topic Map erzeugen kann.

`JavascriptGenerator` bietet die folgende Funktionalität an (vgl. Abbildung 113):

- Erzeugen von JavaScript-Code aus einem `TopicMap`-Objekt, so dass eine authentische, grafische Darstellung der Topic Map mit Hilfe des JavaScripts „`tm_navigate`" berechnet werden kann
- Filterung der Sonderzeichen aus Basenames von Topics und anderen Objekten in der Topic Map, um Probleme der Interpretation der Sonderzeichen zu vermeiden

7.1.4.7.2 Klasse TMTreeNode

`TMTreeNode` bildet die **Grundlage für einen Baum** aus Java-Objekten zur Repräsentation einer Topic Map (vgl. die Beschreibung in Abschnitt 7.1.4.7.3 weiter unten). Jede Instanz von `TMTreeNode` bildet dabei exakt ein Topic aus einer genau definierten Topic Map ab.

Konzeptuell gesehen stellt eine Instanz von `TMTreeNode` den **Knoten eines Baumes** dar. Entsprechend dieser Definition bietet `TMTreeNode` auch die für eine Baum-Datenstruktur typischen Operationen an. Diese umfassen im Detail (vgl. Abbildung 113):

- Erzeugen eines neuen Kind-Knotens mit Initialisierung dessen Attribut-Werte
- Verschieben und Löschen von existierenden Kind-Knoten
- Lesenden Zugriff auf die Kind-Knoten sowie Abfragen der Anzahl der Kind-Knoten
- Zugriff auf den ursprünglichen Inhalt des dem Baum-Knotens zu Grunde liegenden Topics
- Zugriff auf die Associations des dem Baum-Knotens zu Grunde liegenden Topics

7.1.4.7.3 Klasse TMTree

`TMTree` verwendet die Klasse `TMTreeNode`, um einen Baum aus Instanzen von `TMTreeNode` zu bilden. Ein solcher Baum stellt die **Repräsentation** eines Topic Map Graphen **als Spannbaum** (vgl. [Sedge, 1991], S. 475 ff.), ausgehend von einem bei der Erzeugung des Baumes spezifizierten Wurzel-Topics, dar.

`TMTree` und seine Subklassen `TMPartialTree` (vgl. Abschnitt 7.1.4.7.4, Seite 272) und `TMMiniTree` (vgl. Abschnitt 7.1.4.7.5, Seite 273) unterscheiden sich voneinander bezüglich des Algorithmus zur Berechnung des Baumes. Dabei verwendet `TMTree` eine **vollständige Berechnung** aller Baumknoten sofort bei Initialisierung des Baumes. Der Algorithmus kann wie folgt detailliert beschrieben werden:

- Zu Beginn (bei Erzeugung einer Instanz von `TMTree`) spezifiziert der Benutzer das **Wurzel-Topic**. Von diesem ausgehend wird der Baum der Topics (als Spannbaum) vollständig berechnet. Für jedes der Topics, die vom Wurzel-Topic aus er-

reichbar sind, wird eine Instanz von `TMTreeNode` erzeugt und als Knoten in den Baum von `TMTree` eingefügt.

- Wenn der Benutzer zu einem anderen Knoten navigiert, bewegt er sich im Baum entlang der **Kanten** (das sind die *Associations*). Dabei verändert sich die Wurzel des Baums nicht.
- Für Topics, die mit dem ursprünglichen Spannbaum nicht verbunden sind (das sind jene, für die kein Pfad von der Wurzel ausgehend existiert), muss bei Bedarf (d.h. wenn der Benutzer sie als Ziel der Navigation auswählt) ein **vollständig neuer Baum berechnet** werden. In diesem Fall bleiben jedoch die bisher berechneten Bäume erhalten.

Die Klasse `TMTree` bietet folgende Funktionalität zum Zugriff auf den Baum an (vgl. Abbildung 113):

- Berechnung von Iteratoren, die die Knoten des Baumes in verschiedenen Reihenfolgen (*in-order* oder *level-order*) durchwandern
- Zugriff auf einzelne Knoten (wie auch die Wurzel des Baumes) über deren Topic Id
- Zugriff auf die Menge der *Associations*, die in den Knoten des Baumes gespeichert sind

7.1.4.7.4 Klasse TMPartialTree

`TMPartialTree` ist eine Spezialisierung (Subklasse) von `TMTree`. Es unterscheidet sich von seiner Superklasse durch den Algorithmus zur Berechnung des Baumes. `TMPartialTree` berechnet nur eine **Teilmenge des Spannbaums** der zu Grunde liegenden Topic Map. Diese Teilmenge ist wie folgt definiert:

- Zu Beginn wird für das Wurzel-Topic sowie für jedes Topic, das ausgehend vom Wurzel-Topic nicht weiter als zwei Suchschritte entfernt ist, ein Knoten berechnet und in den Baum aufgenommen.
- Verlangt der Benutzer ein Topic, das **weiter als zwei Suchschritte** vom Wurzel-Topic entfernt ist, wird für dieses Topic der entsprechende Baumknoten als Instanz von `TMTreeNode` (vgl. Abschnitt 7.1.4.7.2, Seite 271) berechnet und in den Baum eingefügt. Die Ergebnisse früherer Berechnungen bleiben dabei erhalten.

Die Vorgehensweise zur Berechnung der im Baum enthaltenen Teilmenge aller Topics in einer Topic Map lässt sich somit mit dem **Lazy Swapping Algorithmus** bei der Speicherseiten-Verwaltung in Betriebssystemen vergleichen (siehe L. Borrmann in [RechPom, 1999], S. 647 ff.): neue Teile der Topic Map werden erst geladen, wenn durch einen Zugriff das Fehlen dieser Teile im Speicher bemerkt wird.

Die Funktionalität von `TMPartialTree` ist identisch zu jener seiner Superklasse (vgl. Abbildung 113 und den Abschnitt 7.1.4.7.3 Klasse TMTree, Seite 271).

7.1.4.7.5 Klasse TMMiniTree

`TMMiniTree` ist eine Spezialisierung (Subklasse) von `TMTree`. Es unterscheidet sich von seiner Superklasse durch den Algorithmus zur Berechnung des Baumes. `TMMiniTree` verwendet einen Algorithmus, durch den die Anzahl der **im Baum enthaltenen Knoten** stets **minimal gehalten** wird. Auf diese Weise soll der Aufwand zur Berechnung des Baums auf das Minimum reduziert werden. Der Algorithmus arbeitet wie folgt:

- Zu Beginn wird für das Wurzel-Topic sowie für jedes Topic, das ausgehend vom Wurzel-Topic mit genau einem Suchschritt erreichbar ist, ein Knoten berechnet und in den Baum aufgenommen.
- Navigiert der Benutzer zu einem anderen Topic, so wird dieses Topic das **neue Wurzel-Topic** des Suchbaumes. Wieder werden alle Topics ermittelt, die mit exakt einem Suchschritt von neuem Wurzel-Topic aus erreichbar sind. Für jedes dieser Topics wird erneut eine Instanz von `TMTreeNode` berechnet und als Knoten zum Baum hinzugefügt. Die Ergebnisse **früherer Berechnungen** werden **verworfen** und der Baum vollständig neu berechnet.

Der oben präsentierte Algorithmus setzt auf die These, dass der Aufwand zur Berechnung des minimalen Spannbaumes vernachlässigbar ist. Deshalb kann auf die Bewahrung der Ergebnisse früherer Berechnungen verzichtet werden. Gegenüber den anderen präsentierten Lösungsideen kann ein beträchtlicher Vorteil in Bezug auf die Dauer zur Berechnung des Baumes festgestellt werden.

Die Funktionalität von `TMPartialTree` ist identisch zu jener seiner Superklasse (vgl. Abbildung 113 und den Abschnitt 7.1.4.7.3 Klasse TMTree, Seite 271).

7.2 Klassenmodell des Dokument Splitters

Der Dokument-Splitter dient als **Werkzeug zum Zerlegen** von elektronischen Dokumenten mit dem Ziel, Wissensatome für den Contentpool zu erzeugen. In diesem Abschnitt wird erklärt, wie die Umsetzung der konzeptuellen Architektur des Dokument-Splitters sowie die Implementierung des Splitter-Algorithmus erfolgten.

Zu Beginn dieses Abschnitts werden die Klassen auf oberster Ebene der Klassenhierarchie präsentiert. Die restlichen Klassen des Dokument-Splitters werden entsprechend der Aufteilung in Packages wie folgt behandelt:

7.2.1 Übersicht: oberstes Package „splitter"

Zu Beginn der Präsentation der Implementierung des Dokument-Splitters wird die Übersicht über das gesamte Klassenmodell gezeigt.[36] Im vorliegenden Abschnitt werden anschließend jene Klassen beschrieben, die keinem Sub-Package zugeordnet wurden. Alle diese Klassen befinden sich auf oberster Ebene im Package `jku.ce.scholion.splitter`.

Aus Gründen der leichteren Lesbarkeit wird im gesamten Abschnitt 7.2 auf die ausführliche Angabe des Packagenamens verzichtet. Statt dessen wird immer die kurze Bezeichnung „`splitter`" verwendet. Selbstverständlich befinden sich auch alle Sub-Packages von `splitter` immer innerhalb der Package-Hierarchie `jku.ce.scholion.splitter`.

In Abbildung 114 wird das Klassenmodell der Applikation „Dokument-Splitter" präsentiert. Das Modell zeigt, wie das in Abbildung 61 (Klassenmodell des Dokument-Splitters, siehe Seite 167) präsentierte konzeptuelle Modell umgesetzt wurde.

[36] **Hinweis**: in dieser Übersicht sind auch all jene Packages enthalten, die in den später folgenden Abschnitten noch detailliert behandelt werden.

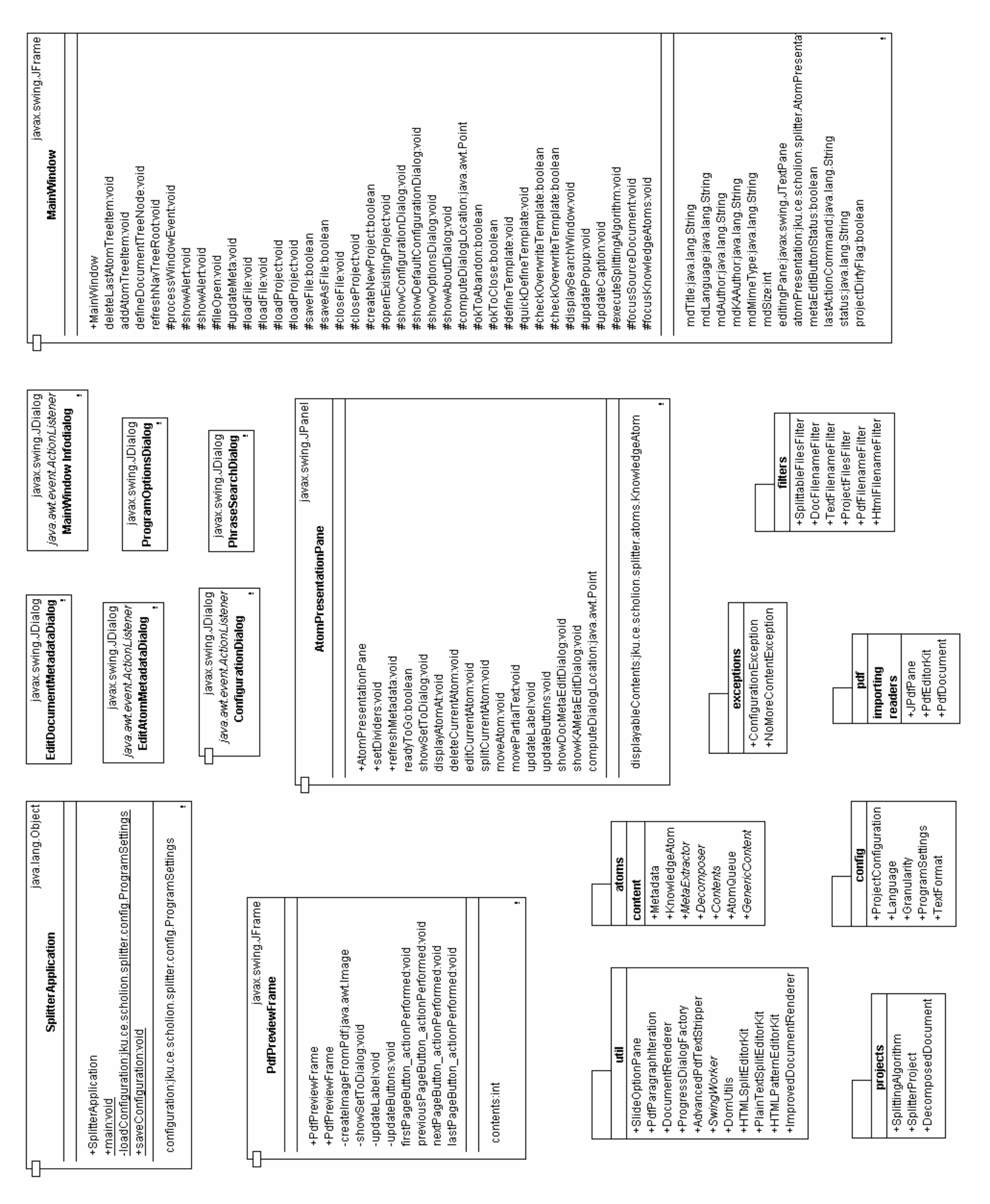


Abbildung 114: Klassenmodell des Dokument-Splitters (Package splitter)

Die Klassen werden nach dem Gesichtspunkt ihrer Aufgaben separat präsentiert, wobei der Autor zunächst auf die Klassen zur Realisierung der grafischen Benutzungsschnittstelle („Graphical User Interface" = GUI) eingeht (vgl. Abschnitt 7.2.1.1 GUI-Klassen, Seite 276 ff.). Danach werden jene Klassen in der obersten Ebene der Klassenhierarchie

erläutert, in welchen die Anwendungslogik zur globalen Steuerung des Programmablaufs implementiert ist (vgl. Abschnitt 7.2.1.2 Klassen zur Implementierung der Anwendungslogik, Seite 278 ff.).

Hinweis: Die Klasse `MainWindow` wird in beiden Abschnitten 7.2.1.1 und 7.2.1.2 vorgestellt. Diese Vorgehensweise wurde deshalb gewählt, weil `MainWindow` weder nur den GUI-Klassen noch ausschließlich den Klassen zur Implementierung der Anwendungslogik zugeordnet werden kann. Vielmehr übernimmt `MainWindow` Aufgaben beiderlei Art. Deshalb werden in beiden Abschnitten die jeweils relevanten Aspekte von `MainWindow` erläutert.

In der Folge werden, wie oben erwähnt, die in der Abbildung 114 gezeigten Klassen auf oberster Ebene der Klassenhierarchie detailliert beschrieben.

7.2.1.1 GUI-Klassen

Die grafische Benutzungsschnittstelle wurde aufgrund der großen **Komplexität der Aufgabenmodelle** (vgl. dazu Abschnitt 6.2 Modellierung der Prozesse im System, Seite 168 ff.) in **mehreren Klassen verteilt** implementiert. Dabei wurden folgende Klassen gebildet:

- Klasse MainWindow (Punkt 7.2.1.1.1)
- Klasse JPdfPane (Punkt 7.2.1.1.2)
- Klasse PdfPreviewFrame (Punkt 7.2.1.1.3)
- Klasse AtomPresentationPane (Punkt 7.2.1.1.4)
- Dialog-Klassen (Punkt 7.2.1.1.5)

7.2.1.1.1 Klasse MainWindow

`MainWindow` erzeugt das **Hauptfenster** der grafischen Benutzungsschnittstelle (GUI) des Dokument-Splitters. Jede Instanz von `SplitterApplication` (vgl. Punkt 7.2.1.2.1) erhält genau eine Instanz von `MainWindow`, das als **Container** der obersten Ebene für alle **weiteren Komponenten** der GUI dient.

`MainWindow` stellt dem Benutzer folgende Möglichkeiten zur Verfügung (vgl. Abbildung 114):

- Je einen Bildschirm-Bereich für die Navigation durch **Projekte** (links), das Betrachten von **Dokumenten** bzw. Wissensatomen (zentral oben) und für das Betrachten von **Wissensatomen** (zentral unten).
- Eine **Menüleiste** enthält Menüs mit Menü-Buttons, die zur einfachen Bedienung des Dokument-Splitters beitragen.
- Am unteren Rand der GUI wird zudem eine **Statusleiste** angezeigt, die Auskunft über den aktuellen Status der Applikation gibt.

Die auf Aufgaben bezogene Funktionalität von `MainWindow` wird später im Punkt 7.2.1.2.2 (siehe Seite 278) behandelt.

7.2.1.1.2 Klasse JPdfPane

Hinweis: Auch die Klasse `JPdfPane` erfüllt Aufgaben der grafischen Benutzungsschnittstelle. Da `JPdfPane` jedoch in einem hohen Ausmaß an der Darstellung von Inhalten im PDF-Format beteiligt ist, wurde es dem Package „`pdf`“ zugeordnet. Der Benutzer wird dazu auf den Abschnitt 7.2.8 (`JPdfPane` wird in 7.2.8.1, Seite 299 behandelt) verwiesen.

7.2.1.1.3 Klasse PdfPreviewFrame

`PdfPreviewFrame` erzeugt von jedem vom Benutzer geöffneten Dokument im PDF-Format eine **Vorschau**, welche anschließend in einem **eigenen Fenster** (unabhängig von der aktuellen `MainWindow`-Instanz) angezeigt wird. Die Implementierung einer solchen Klasse zur Vorschau von PDF Dokumenten ist deshalb notwendig, da die Extraktion eines Texts eine andere Darstellung am Bildschirm ergibt als das Original-Dokument.

Hinweis: Die Extraktion von Text aus PDF-Dokumenten ist eine algorithmisch komplexe Aufgabe. Welche Schwierigkeiten dabei auftreten, wird im Abschnitt 9.2 (Future Work, Seite 319 f.) beschrieben.

Folgende Möglichkeiten werden von `PdfPreviewFrame` angeboten (vgl. Abbildung 114):

- Anzeigen einer bestimmten Seite aus dem aktuell geöffneten Dokument im PDF-Format
- Navigieren durch die Seiten des Dokuments mit Hilfe der an der GUI angebotenen Buttons

7.2.1.1.4 Klasse AtomPresentationPane

Die Klasse `AtomPresentationPane` ist spezialisiert auf die **Darstellung von Wissensatomen**. Dabei wird unterschieden zwischen den Wissensatomen an sich und den Metadaten dazu. Die Wissensatome werden im oberen Bereich, die **Metadaten** im unteren Bereich der Benutzungsschnittstelle gezeigt. Beide Teile können vom Benutzer auf Wunsch ausgeblendet werden.

`AtomPresentationPane` bietet dem Benutzer folgende Möglichkeiten (vgl. Abbildung 114):

- Betrachten eines Wissensatoms und der Metadaten hierzu
- Navigieren durch die Menge der gegenwärtig verfügbaren Wissensatome mit Hilfe von an der GUI zur Verfügung gestellten Buttons
- Bearbeiten der Wissensatome (Verschmelzen, Zerteilen, Löschen, Editieren) mit Hilfe von Buttons

7.2.1.1.5 Dialog-Klassen

Mehrere Klassen werden benötigt, um Dialoge zu implementieren, die in **Kommunikation mit dem Benutzer** die für einen bestimmten Anwendungsfall notwendige Entscheidungen oder Eingabedaten ermitteln. Folgende Dialog-Klassen werden von Dokument-Splitter verwendet (mit je einer kurzen Erklärung – vgl. Abbildung 114):

- ProgramOptionsDialog: dient zur Konfiguration globaler Parameter der Applikation „Dokument-Splitter".
- ConfigurationDialog: dient zur Konfiguration von Parametern, die je für ein Projekt gültig sind.
- PhraseSearchDialog: öffnet ein Fenster, das dem Benutzer erlaubt, innerhalb des Textes eines geöffneten Dokuments nach Schlagwörtern zu suchen. Beim Suchvorgang können mehrere Optionen (z.B. Groß- und Kleinschreibung beachten / nicht beachten) spezifiziert werden.
- MainWindow_Infodialog: öffnet eine Dialogbox, die Information zur Applikation und den Hinweis (Hyperlink) auf die Homepage des Projekts Scholion WB+ (http://scholion.ce.jku.at) enthält.
- EditAtomMetadataDialog: erlaubt dem Benutzer das Bearbeiten von Metadaten, die auf Ebene eines Wissensatoms gelten (z.B. Stichwort(e), Anmerkung).
- EditDocumentMetadataDialog: erlaubt dem Benutzer das Bearbeiten von Metadaten, die je für ein gesamtes Dokument gültig sind (z.B. Name des Autors, Quelle als URL).

7.2.1.2 Klassen zur Implementierung der Anwendungslogik

Im Vergleich zur grafischen Benutzungsschnittstelle (deren Klassen eben im Abschnitt 7.2.1.1, Seite 276 ff. erläutert wurden) ist die Komplexität der Anwendungslogik für die globale Steuerung der Applikation „Dokument-Splitter" gering. Die Aufgaben für die Steuerung des Dokument-Splitters wurden in folgenden zwei Klassen implementiert:

- Klasse SplitterApplication (Punkt 7.2.1.2.1)
- Klasse MainWindow (Punkt 7.2.1.2.2)

7.2.1.2.1 Klasse SplitterApplication

SplitterApplication steuert den **Kontrollfluss der Applikation** „Dokument-Splitter" auf oberster Ebene. Beim Start des Dokument-Splitters wird immer zuerst eine Instanz von SplitterApplication erzeugt. Während der Initialisierung sorgt die Instanz dafür, dass die GUI geladen und richtig initialisiert wird.

Die Klasse SplitterApplication erfüllt folgende Aufgaben (vgl. Abbildung 114):

- Laden der Konfiguration des Dokument-Splitters, welche in einem Objekt der Klasse ProgramSettings (vgl. dazu Abschnitt 7.2.2.1, Seite 280) gespeichert ist
- Erzeugen und Initialisieren je einer Instanz von MainWindow und AtomPresentationPane

7.2.1.2.2 Klasse MainWindow

Nach der Initialisierung der Applikation „Dokument-Splitter" durch SplitterApplication erhält MainWindow zu einem großen Teil die Kontrolle über den **Programmablauf**. Diese Architektur wurde deshalb gewählt, da in MainWindow auch die gesamten Komponenten

der GUI enthalten sind. Somit ist die „Distanz zwischen Benutzer und Programm" in der Klasse `MainWindow` minimal.

`MainWindow` stellt dem Benutzer folgende Funktionalität (aus der Sichtweise der Anwendungslogik betrachtet) zur Verfügung (vgl. Abbildung 114):

- Erstellen, Öffnen und Schließen von **Projekten**
- Konfiguration der **Parameter** der Applikation oder eines Projekts
- Öffnen und Schließen von **Dokumenten** in einem Projekt
- Durchführen der **Zerlegung** eines Dokuments in Wissensatome
- Wechseln der **Ansicht** vom Quelldokument zu den Wissensatomen (falls vorhanden), und umgekehrt

Zusätzlich ist `MainWindow` für die Weiterleitung all jener Information an `SplitterApplication` verantwortlich, welche für die Applikation global von Bedeutung ist. Dieser Fall tritt z.B. ein, wenn der Benutzer die globalen Parameter rekonfiguriert.

7.2.2 Package „config"

Im Package `splitter.config` sind alle Klassen zusammen gefasst, die zur Speicherung von **Zuständen bzw. Konfigurationen** eines Projekts oder der Applikation Dokument-Splitter dienen.

Zu diesem Zweck werden die folgenden, in den in Klammern genannten Abschnitten beschriebenen Klassen benötigt:

- Klasse ProgramSettings (Abschnitt 7.2.2.1)
- Klasse ProjectConfiguration (Abschnitt 7.2.2.2)
- Klasse Language (Abschnitt 7.2.2.3)
- Klasse Granularity (Abschnitt 7.2.2.4)
- Klasse TextFormat (Abschnitt 7.2.2.5)

Abbildung 115 gibt einen Überblick über die oben genannten Teile des Packages `splitter.config`.

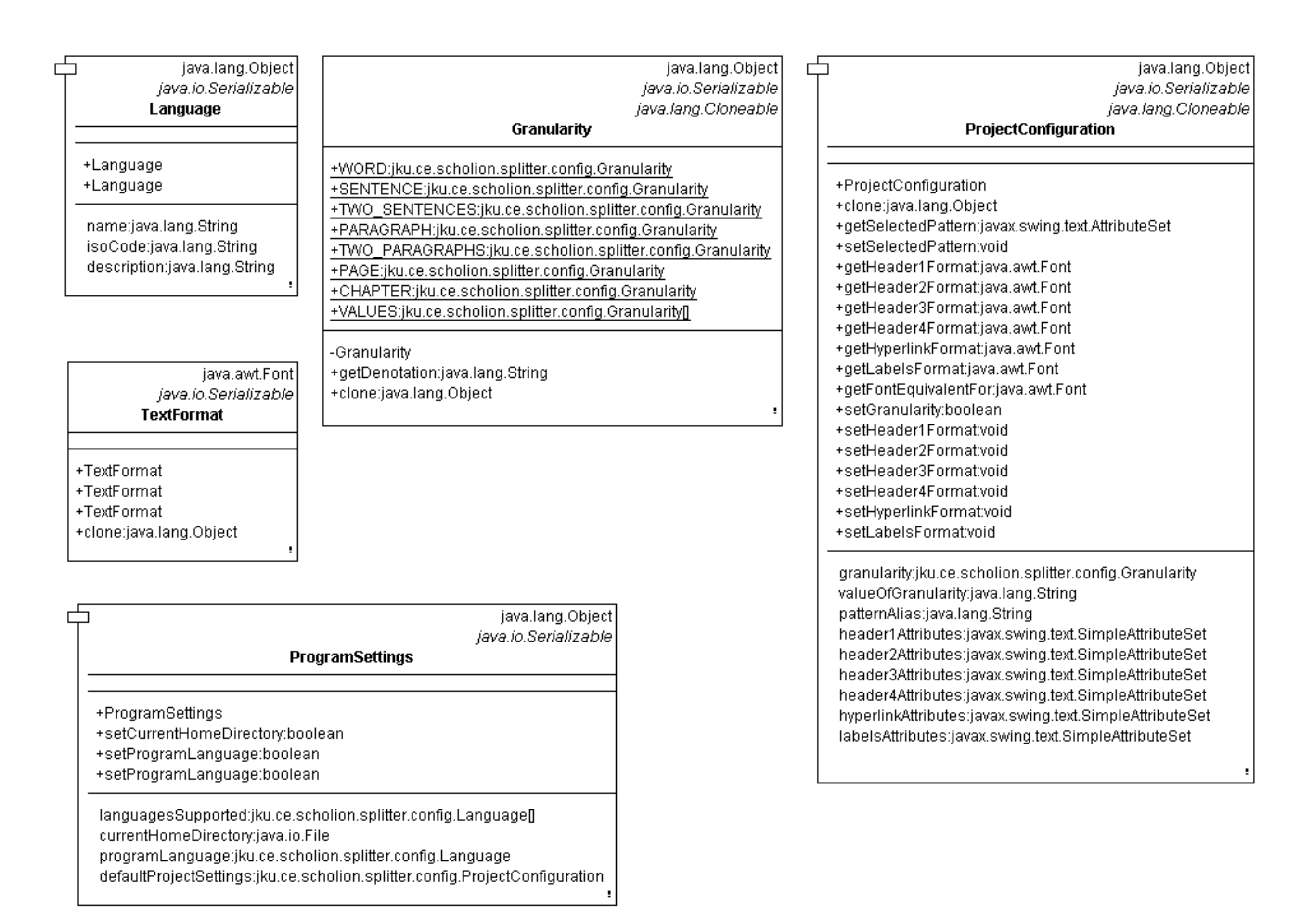


Abbildung 115: Klassenmodell – Package splitter.config

7.2.2.1 Klasse ProgramSettings

`ProgramSettings` enthält Objekte zur Beschreibung der vom Benutzer konfigurierten Parameter für die **globale Steuerung** des Dokument-Splitters. Pro Instanz des Dokument-Splitters gibt es nur ein Objekt der Klasse `ProgramSettings`.

Die Klasse `ProgramSettings` stellt get()- und set()-Methoden für folgende globale Eigenschaften (Parameter) des Dokument-Splitters zur Verfügung (vgl. Abbildung 115):

- Arbeitsverzeichnis („Home directory")
- Gewählte Sprache (wobei im gegenwärtigen Stadium der Implementierung jedoch nur die Standardsprache „Deutsch" unterstützt wird)
- Vorgabe-Einstellungen für neu angelegte Projekte

7.2.2.2 Klasse ProjectConfiguration

`ProjectConfiguration` enthält die Objekte zur Beschreibung der Ausprägungen der Parameter, welche vom Benutzer für jeweils ein **Projekt** definiert werden und **lokale Einstellungen** konfigurieren.

Die Klasse ProjectConfiguration stellt get()- und set()-Methoden für folgende lokale Eigenschaften (Parameter) eines Projekts zur Verfügung (vgl. Abbildung 115):

- Gewünschte **Granularität** der von diesem Projekt erzeugten Wissensatome (als Instanz der Klasse Granularity, vgl. Abschnitt 7.2.2.4, Seite 281)
- **Muster-Vorlagen** für die Formatierung folgender Objekte in einem Dokument: Überschriften der Gliederungs-Ebenen 1 bis 4; Beschriftungen von Abbildungen und ähnlichen Objekten; Querverweise innerhalb eines Dokuments, oder Hyperlinks.

7.2.2.3 Klasse Language

Language dient der **Modellierung einer Sprache**, die durch den Dokument-Splitter unterstützt wird. Der Begriff „unterstützen" meint in diesem Zusammenhang, dass von Dokument-Splitter die Elemente der GUI in einer bestimmten Sprache zur Verfügung gestellt werden.

Folgende **Eigenschaften** einer Sprache werden durch die Klasse Language abgebildet (vgl. Abbildung 115):

- Ausführliche Bezeichnung der Sprache
- ISO-Kürzel der Sprache
- Optionale Beschreibung für die modellierte Sprache

7.2.2.4 Klasse Granularity

Granularity ist eine Enumerations-Klasse. Von Granularity können folglich keine Instanzen angelegt werden. Die in Granularity vorgegebenen Werte modellieren die zur Zeit im Dokument-Splitter unterstützten Möglichkeiten zur **Steuerung der Größe** von aus Dokumenten erzeugten Wissensatomen.

Folgende Einstellungen für die Granularität werden gegenwärtig von Granularity unterstützt (vgl. Abbildung 115):

- Ein Wort
- Ein Satz oder zwei Sätze
- Ein oder zwei Absätze
- Eine ganze Seite
- Ein ganzes Kapitel

Granularity bietet für jede dieser Einstellungen zudem eine textuelle Beschreibung an.

7.2.2.5 Klasse TextFormat

TextFormat ist eine Erweiterung der Java-Klasse java.awt.Font dahingehend, dass das Interface java.lang.Serializable erfüllt wird. Dadurch können Objekte der Klasse TextFormat unter anderem in Dateien geschrieben und von Dateien gelesen werden.

Die Funktionalität von TextFormat ist gegenüber java.awt.Font nicht verändert. Eine Instanz dieser Klasse ist für die Anzeige von Text in einer bestimmten, genau definierten Schriftart verantwortlich.

7.2.3 Package „exceptions“

Das Package splitter.exceptions enthält Objekte, die bestimmte, **vordefinierte Ausnahmesituationen** während der Ausführung des Dokument-Splitters modellieren. Es besteht aus folgenden, in den in Klammer angegebenen Abschnitten beschriebenen, Klassen:

- Klasse ConfigurationException (Abschnitt 7.2.3.1)
- Klasse NoMoreContentException (Abschnitt 7.2.3.2)

Abbildung 116 gibt einen Überblick über die oben genannten Teile des Packages splitter.exception.

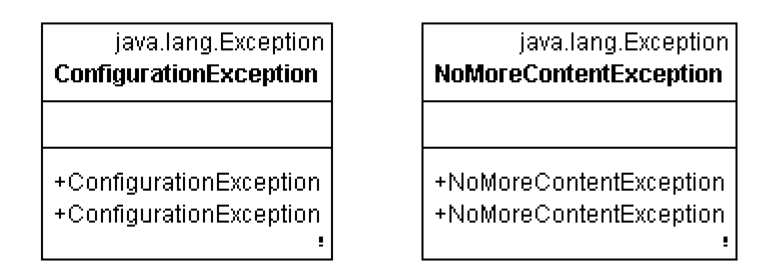


Abbildung 116: Klassenmodell – Package splitter.exceptions

7.2.3.1 Klasse ConfigurationException

Eine ConfigurationException zeigt Ausnahmesituationen an, in denen eine Klasse des Dokument-Splitters **nicht korrekt initialisiert oder konfiguriert** wurde, bevor seine Dienste in Anspruch genommen wurden. Dies könnte z.B. passieren, wenn die Konfiguration eines Projekts nicht geladen werden kann (vgl. Punkt 7.2.1.2.1, Seite 278).

7.2.3.2 Klasse NoMoreContentException

Eine NoMoreContentException zeigt Ausnahmesituationen an, in denen versucht wurde, aus einem Dokument noch Daten zu extrahieren, obwohl keine mehr vorhanden waren. Dies kann z.B. bei der Verwendung eines Objekts der Klasse PdfParagraphIteration (vgl. Abschnitt 7.2.4.3, Seite 285) oder bei einem Objekt, das das Interface Decomposer erfüllt (vgl. 7.2.7.4, Seite 294), passieren.

7.2.4 Package „util“

Im Package splitter.util sind wichtige **Werkzeug-Klassen** zusammen gefasst, die für die Ausführung des Dokument-Splitters benötigt werden. Es besteht aus folgenden, in den in Klammer angegebenen Abschnitten detailliert beschriebenen, Klassen:

- Klasse DocumentRenderer (Abschnitt 7.2.4.1)

- Klasse ImprovedDocumentRenderer (Abschnitt 7.2.4.2)
- Klasse PdfParagraphIteration (Abschnitt 7.2.4.3)
- Klasse AdvancedPdfTextStripper (Abschnitt 7.2.4.4)
- Klasse SlideOptionPane (Abschnitt 7.2.4.5)
- Klasse ProgressDialogFactory (Abschnitt 7.2.4.6)
- Klasse SwingWorker (Abschnitt 7.2.4.7)
- Klasse DomUtils (Abschnitt 7.2.4.8)
- Klasse PlainTextSplitEditorKit (Abschnitt 7.2.4.9)
- Klasse HTMLSplitEditorKit (Abschnitt 7.2.4.10)
- Klasse HTMLPatternEditorKit (Abschnitt 7.2.4.11)

Abbildung 117 gibt einen Überblick über die oben genannten Teile des Packages `splitter.util`.

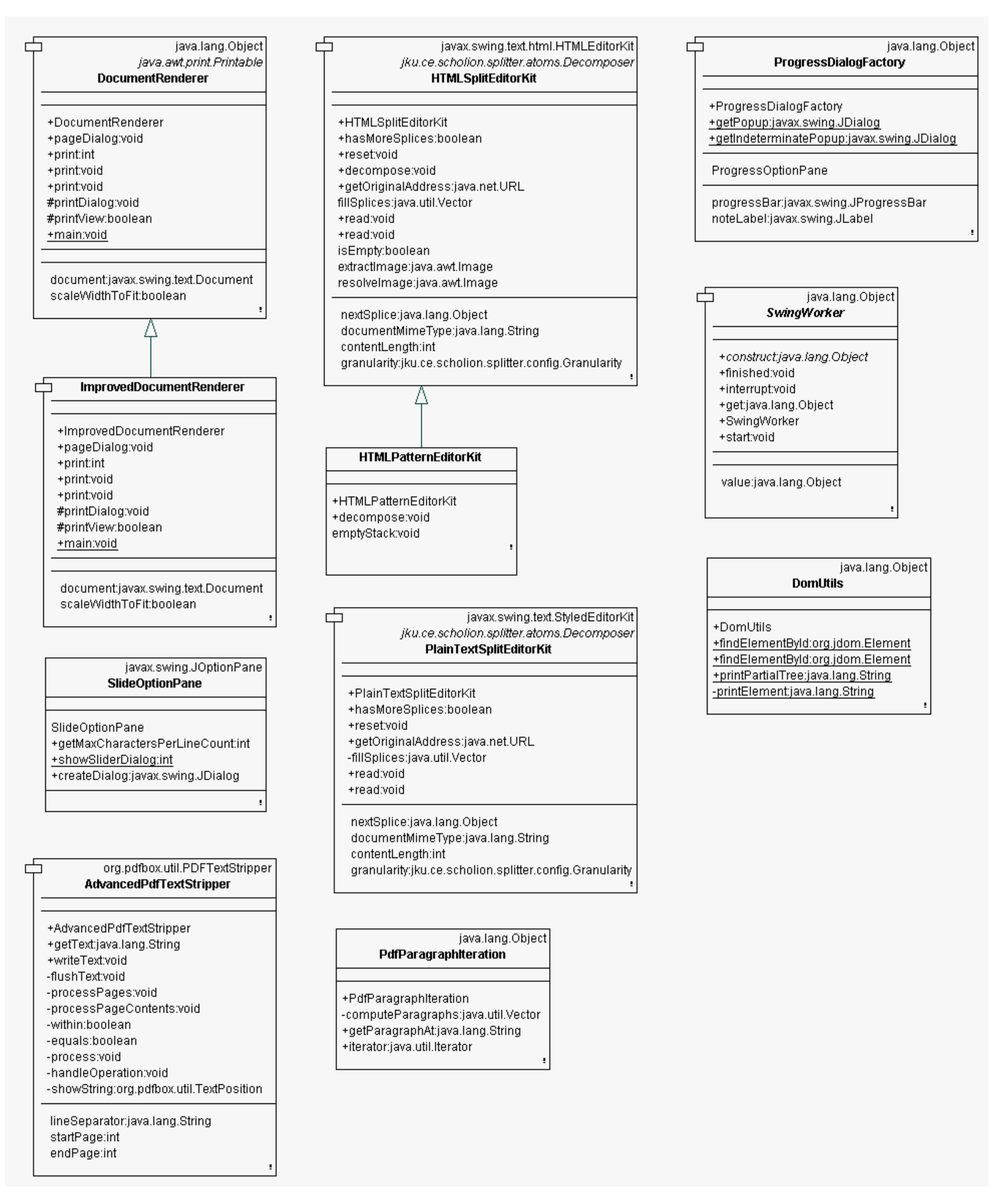


Abbildung 117: Klassenmodell – Package splitter.util

7.2.4.1 Klasse DocumentRenderer

`DocumentRenderer` ist eine Klasse, mit deren Hilfe die **Darstellung eines HTML-Dokuments** in der grafischen Benutzungsschnittstelle des Dokument-Splitters gedruckt wer-

den kann. Es wurde auf der Grundlage von [ExpExch, 2003] entwickelt. Bei der Erstellung des Druckauftrages werden alle wichtigen Formatierungen des Textes berücksichtigt.

Die Klasse `DocumentRenderer` wurde vor allem zur Erleichterung der Entwicklung benötigt. Für die derzeitige Version des Dokument-Splitters ist `DocumentRenderer` nur von untergeordneter Bedeutung. Daher gehe ich hier nicht näher darauf ein.

7.2.4.2 Klasse ImprovedDocumentRenderer

`ImprovedDocumentRenderer` ist eine Subklasse von `DocumentRenderer` (siehe oben). Darin wurden einige Algorithmen für die Berechnung des Druckauftrages korrigiert. Ansonsten ist die Funktionalität identisch mit jener von `DocumentRenderer`.

7.2.4.3 Klasse PdfParagraphIteration

`PdfParagraphIteration` liefert aus dem Text eines Dokuments im PDF-Format eine Iteration durch jene Elemente, die auf Grundlage einer `Granularity`-Konfiguration erstellt wurden. Die Konfiguration der Granularität erfolgt durch die Übergabe eines `Granularity`-Objekts (vgl. Abschnitt 7.2.2.4, Seite 281) an die Schnittstelle von `PdfParagraphIteration`. Die Iteration über die Elemente des PDF-Dokuments stellt folgende Funktionalität zur Verfügung (vgl. Abbildung 117):

- Direkter Zugriff auf ein Element über seinen numerischen Index (fortlaufende Nummer innerhalb des Dokuments)
- Navigation durch die Elemente mit Hilfe des Iterators

7.2.4.4 Klasse AdvancedPdfTextStripper

`AdvancedPdfTextStripper` ist eine Klasse, die der **Extraktion von reinem Text** (ohne Information über die Formatierung) aus Dokumenten im PDF-Format dient. Eine solche Klasse wird notwendig, da das Lesen von Text aus PDF-Dokumenten eine alles andere als triviale Aufgabe darstellt.

`AdvancedPdfTextStripper` basiert auf der Klasse `org.pdfbox.util.PDFTextStripper` [PdfBox, 2002]. Es wurde auf die Bedürfnisse des Dokument-Splitters optimiert und stellt folgende Funktionalität zur Verfügung (vgl. Abbildung 117):

- Erzeugen einer neuen Instanz auf Basis eines bereits geparsten Dokuments im PDF-Format
- Spezifikation einer „von-Seite" und einer „bis-Seite" – d.h. Spezifikation eines Teilbereichs im Dokument, aus dem der reine Text extrahiert werden soll
- Zugriff auf den fertig berechneten, reinen Text

7.2.4.5 Klasse SlideOptionPane

`SlideOptionPane` bildet die Grundlage für **Dialoge**, die dem Benutzer ermöglichen, mit Hilfe eines **Auswahlbalkens** einen Wert zwischen einem vordefinierten Minimum und

Maximum auszuwählen. Instanzen von `SlideOptionPane` kommen bei folgenden Gelegenheiten zum Einsatz:

- Auswahl des anzuzeigenden Wissensatoms (vgl. Punkt 7.2.1.1.4 Klasse AtomPresentationPane, Seite 277)
- Auswahl jener Seite eines PDF-Dokuments, deren Vorschau angezeigt werden soll (vgl. Punkt 7.2.1.1.3 Klasse PdfPreviewFrame, Seite 277)

7.2.4.6 Klasse ProgressDialogFactory

Die Werkzeug-Klasse `ProgressDialogFactory` dient zur Erzeugung von standardisierten Popups, die den **Fortschritt einer Operation** oder eines Vorgangs anzeigen, der einen längeren Zeitraum bis zu seiner Fertigstellung in Anspruch nimmt (z.B. die Zerlegung eines Dokuments in Wissensatome).

`ProgressDialogFactory` bietet die folgende Funktionalität an (vgl. Abbildung 117):

- Erzeugen eines Popups, das den Fortschritt eines längeren Vorgangs exakt anzeigt
- Erzeugen eines Popups, das den Fortschritt eines längeren Vorgangs nicht exakt anzeigt, aber erkennen lässt, dass das Programm noch aktiv ist

7.2.4.7 Klasse SwingWorker

Die Klasse `SwingWorker` dient zur effizienteren **Ausführung von Aufgaben in Applikationen**, indem es längere Operationen in eine **eigene Prozess-Warteschlange** der Java Virtual Machine auslagert [JavaTut, 2003].

Alle Aufgaben, die von einem `SwingWorker` erfüllt werden sollen, müssen in einer anonymen Subklasse deklariert werden. Durch Aufruf der `start()`-Methode wird die Ausführung des zuvor spezifizierten Codes ausgelöst.

7.2.4.8 Klasse DomUtils

`DomUtils` schließt Lücken, die in der Implementierung des **Document Object Model** (DOM) für Java existieren und definiert zusätzliche nützliche Operationen.

Folgende Funktionalität wird von `DomUtils` als Werkzeugklasse angeboten (vgl. Abbildung 117):

- Auffinden eines Elements in einem DOM-Baum durch seinen eindeutigen Identifier
- Umwandlung eines einzelnen Elements des DOM-Baums in eine Zeichenkette
- Umwandlung eines Teilbaums aus dem DOM-Baum in eine Zeichenkette

7.2.4.9 Klasse PlainTextSplitEditorKit

`PlainTextSplitEditorKit` ist eine Hilfsklasse, die eine **Schnittstelle** auf hohem Abstraktionsniveau für die **Zerlegung von reinen Text-Dokumenten** anbietet. Folgende Funktionalität wird von `PlainTextSplitEditorKit` unterstützt (vgl. Abbildung 117):

- Abfrage der Gesamtlänge des Dokuments
- Zugriff auf das nächste, durch eine Instanz von `Granularity` (vgl. Abschnitt 7.2.2.4, Seite 281) in seiner Länge definierte Teilstück aus dem Dokument
- Abfrage, ob ein weiteres Teilstück noch existiert
- Zugriff auf das Originaldokument in Form seiner URL

7.2.4.10 Klasse HTMLSplitEditorKit

`HTMLSplitEditorKit` ist eine Hilfsklasse, die eine Schnittstelle auf hohem Abstraktionsniveau für die **Zerlegung von HTML-Dokumenten** anbietet. Folgende Funktionalität wird von `HTMLSplitEditorKit` angeboten (vgl. Abbildung 117):

- Abfrage der Gesamtlänge des HTML-Dokuments
- Zugriff auf das nächste, durch eine Instanz von `Granularity` (vgl. Abschnitt 7.2.2.4, Seite 281) in seiner Länge definierte Teilstück aus dem HTML-Dokument
- Abfrage, ob ein weiteres Teilstück noch existiert
- Zugriff auf das Originaldokument in Form seiner URL

7.2.4.11 Klasse HTMLPatternEditorKit

`HTMLPatternEditorKit` bietet wie seine Superklasse `HTMLSplitEditorKit` eine Schnittstelle auf hohem Abstraktionsniveau für die **Zerlegung von HTML-Dokumenten**. Der Algorithmus zur Zerlegung ist jedoch, im Gegensatz zu seiner Superklasse, auf Grundlage der Spezifikation einer **Muster-Formatierung** an Statt einer Instanz von `Granularity` implementiert.

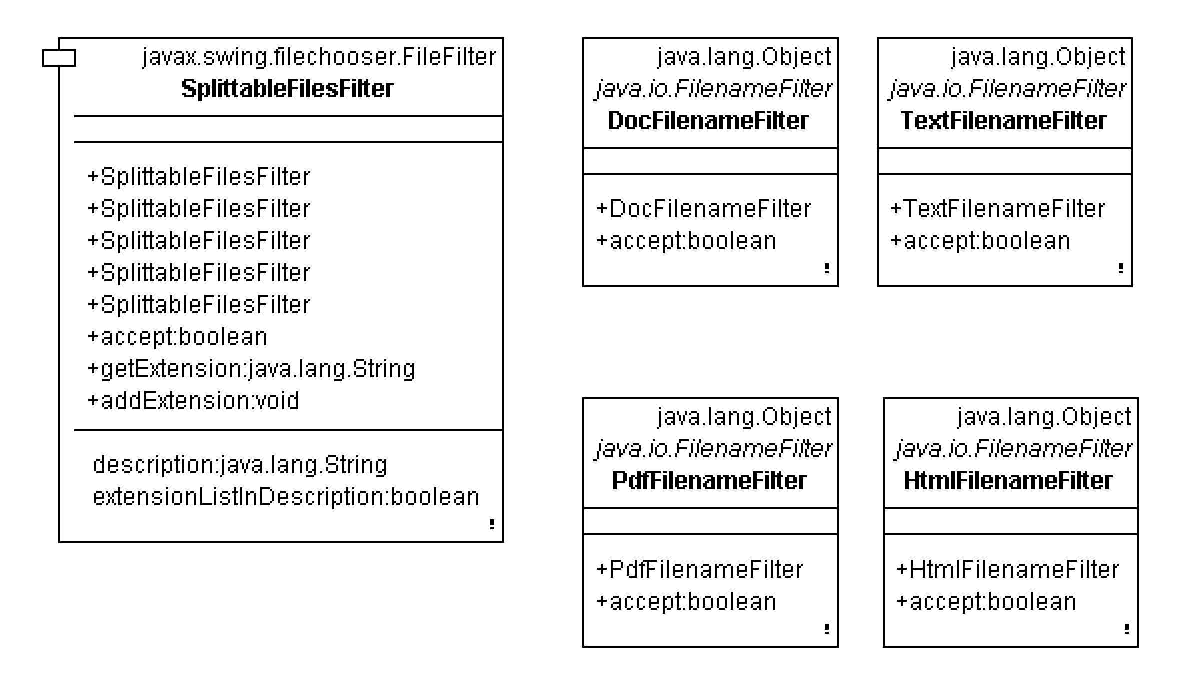
Die Schnittstelle, d.h. die von `HTMLPatternEditorKit` angebotene Funktionalität, unterscheidet sich nicht von jener seiner Superklasse (vgl. 7.2.4.10 oben).

7.2.5 Package „filters“

Das Package `splitter.filters` enthält Klassen, die zur **Filterung** von Dateien im lokalen **Dateisystem** nach verschiedenen Kriterien dienen. Auf diese Weise können Dateien leichter gefunden werden, die für den Dokument-Splitter relevant sind. Das Package besteht aus folgenden, in den in Klammern genannten Abschnitten detailliert beschriebenen Klassen:

- Klasse ProjectFilesFilter (Abschnitt 7.2.5.1)
- Klasse SplittableFilesFilter (Abschnitt 7.2.5.2)
- Klasse TextFilenameFilter (Abschnitt 7.2.5.3)
- Klasse DocFilenameFilter (Abschnitt 7.2.5.4)
- Klasse HtmlFilenameFilter (Abschnitt 7.2.5.5)
- Klasse PdfFilenameFilter (Abschnitt 7.2.5.6)

Abbildung 118 gibt einen Überblick über die oben genannten Teile des Packages `splitter.filters`.



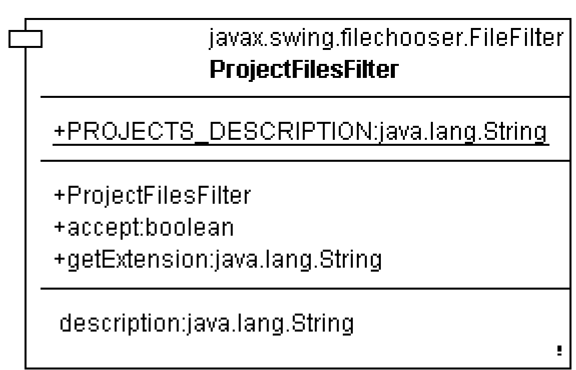


Abbildung 118: Klassenmodell – Package splitter.filters

7.2.5.1 Klasse ProjectFilesFilter

`ProjectFilesFilter` dient zum Filtern der Dateien im lokalen Dateisystem mit dem Zweck, Dateien zu finden, die **Projekte des Dokument-Splitters** enthalten. Projektdateien des Dokument-Splitters können an der **Datei-Endung „.spr"** erkannt werden. Weiters enthält `ProjectFilesFilter` eine textuelle Beschreibung für die Projektdateien.

Verwendung findet `ProjectFilesFilter` in Verbindung mit der Klasse `MainWindow` (vgl. Punkt 7.2.1.1.1, Seite 276). Er tritt beim Öffnen von vorhandenen Projektdateien in Aktion.

7.2.5.2 Klasse SplittableFilesFilter

SplittableFilesFilter dient zum Filtern der Dateien im lokalen Dateisystem mit dem Zweck, Dateien zu finden, die vom Dokument-Splitter (im Rahmen eines Projekts) zu Wissensatomen zerlegt werden können. In SplittableFilesFilter werden alle Dateien angezeigt, die einem der folgenden Formate entsprechen:

- Reine Textdatei: „.txt", „.wri" (MIME-Type: text/plain)
- HTML Datei: „.htm", „.html" (MIME-Type: text/html)
- PDF-Datei: „.pdf" (MIME-Type: application/pdf)

Die Unterstützung für Microsoft-Word-Dateien („.doc"; MIME-Type: application/ms-word) ist erst in einem prototypischen Ausmaß vorhanden. Deshalb werden Word-Dateien zur Zeit mit dem SplittableFilesFilter noch nicht angezeigt.

Verwendung findet SplittableFilesFilter in Verbindung mit der Klasse MainWindow (vgl. Punkt 7.2.1.1.1, Seite 276). Er tritt beim Öffnen eines Dokuments für ein Projekt in Aktion.

7.2.5.3 Klasse TextFilenameFilter

TextFilenameFilter verfolgt den Zweck, im lokalen Dateisystem Dateien aufzufinden, die **reinen Text** enthalten. Er sucht nach den Datei-Endungen „.txt" und „.wri".

7.2.5.4 Klasse DocFilenameFilter

DocFilenameFilter verfolgt den Zweck, im lokalen Dateisystem Dateien aufzufinden, die Dokumente im **Microsoft Word Format** enthalten. Er sucht nach der Datei-Endung „.doc".

7.2.5.5 Klasse HtmlFilenameFilter

HtmlFilenameFilter verfolgt den Zweck, im lokalen Dateisystem Dateien aufzufinden, die **Dokumente im HTML-Format** enthalten. Er sucht nach den Datei-Endungen „.htm" und „.html".

7.2.5.6 Klasse PdfFilenameFilter

PdfFilenameFilter verfolgt den Zweck, im lokalen Dateisystem Dateien aufzufinden, die **Dokumente im „Portable Document Format"** (PDF) Format enthalten. Er sucht nach der Datei-Endung „.pdf".

7.2.6 Package „projects"

Im Package splitter.projects sind Klassen enthalten, die Daten über Projekte der Benutzer von Dokument-Splitter verwalten oder erzeugen. Es besteht aus folgenden, in den in Klammern angegebenen Abschnitten detailliert beschriebenen Klassen:

- Klasse SplitterProject (Abschnitt 7.2.6.1)

- Klasse SplittingAlgorithm (Abschnitt 7.2.6.2)
- Klasse DecomposedDocument (Abschnitt 7.2.6.3)

Abbildung 119 gibt einen Überblick über die oben genannten Teile des Packages `splitter.projects`.

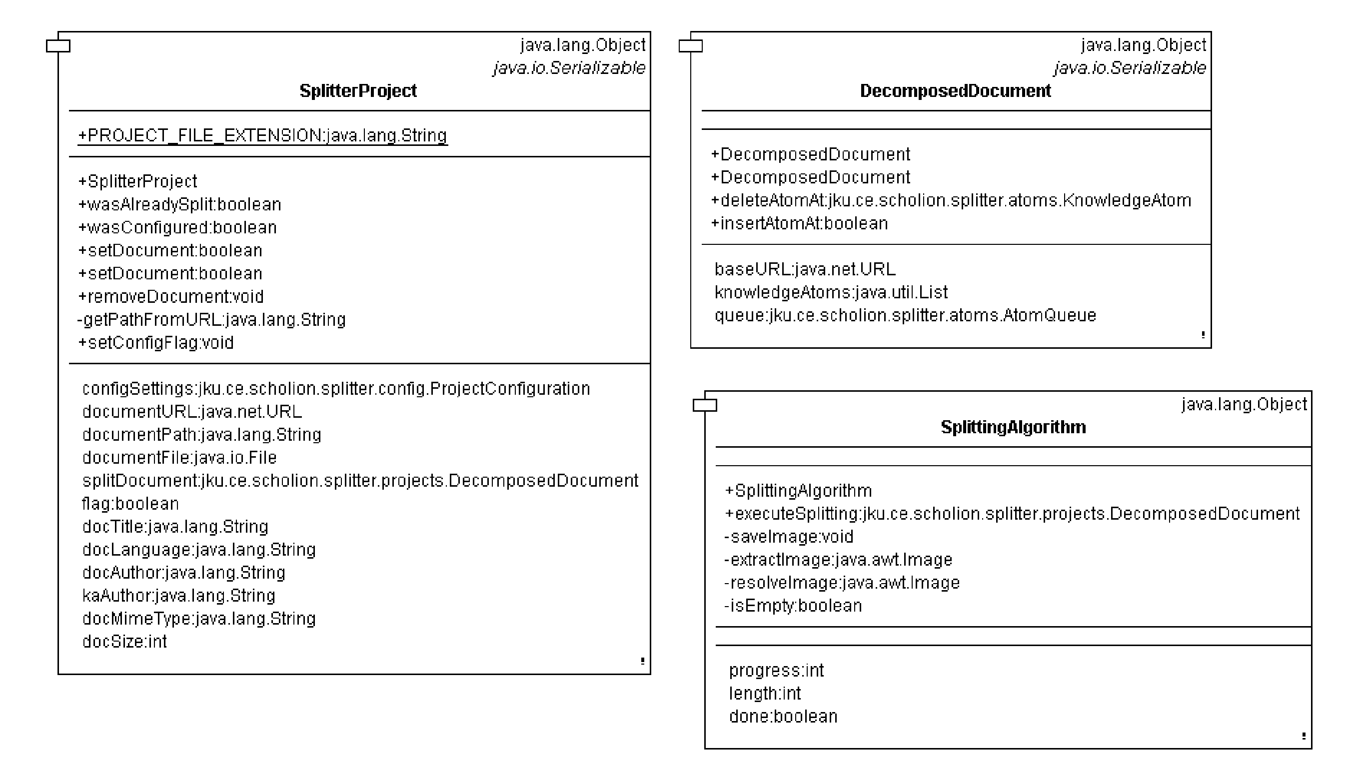


Abbildung 119: Klassenmodell – Package splitter.projects

7.2.6.1 Klasse SplitterProject

`SplitterProject` modelliert alle **Fakten über ein Projekt** eines Benutzers von Dokument-Splitter in Java-Objekten. Eine Instanz von `SplitterProject` enthält die folgende Information:

- Eine Instanz von `ProjectConfiguration` (vgl. Abschnitt 7.2.2.2, Seite 280), in der die **Konfiguration** der Optionen für das Projekt gespeichert ist.
- Optional ein **Dokument**, das die Grundlage für die vom Dokument-Splitter im Projekt erzeugten oder zu erzeugenden Wissensatome darstellt. Jedes Projekt kann maximal 1 Dokument enthalten; d.h. pro zu zerlegendem Dokument muss es eine eigene Instanz von `SplitterProject` geben. Im Projekt wird nicht das Dokument selbst gespeichert, sondern lediglich seine Adresse in Form einer URL.
- Eine beliebige **Anzahl von Wissensatomen**, die aus dem Dokument des Projekts erzeugt wurden. Selbstverständlich ist Voraussetzung für das Vorhandensein von Wissensatomen, dass für das Projekt bereits ein Dokument ausgewählt wurde. Wenn noch kein Dokument ausgewählt oder der Zerlegungsvorgang noch nicht gestartet wurde, enthält ein Projekt keine Wissensatome. Nach dem Start

der Zerlegung kann ein Projekt eine beliebig große Anzahl an Wissenatomen enthalten, die in einem Objekt der Klasse `DecomposedDocument` (vgl. Abschnitt 7.2.6.3, Seite 291) gespeichert und verwaltet werden.

`SplitterProject` bietet für die oben genannte Information für jedes Objekt je eine `get()`- und eine `set()`-Methode an. Zusätzlich existieren noch **Zugriffsmethoden** für folgende Metadaten des Dokuments, sofern dieses bereits definiert (ausgewählt) wurde (vgl. Abbildung 119):

- Autor des Dokuments
- Titel des Dokuments
- Sprache, in der das Dokument verfasst wurde
- Name des Verfassers der Wissensatome (d.h. Name des Benutzers, von dem das Projekt erstellt wurde)
- MIME-Type des Dokuments

7.2.6.2 Klasse SplittingAlgorithm

In `SplittingAlgorithm` ist der **Algorithmus zur Zerlegung von Dokumenten** implementiert. Die Schnittstelle von `SplittingAlgorithm` ist sehr einfach zu verwenden. Das **geöffnete Dokument** wird übergeben. Anschließend wird die Zerlegung umgehend gestartet. So bald die Zerlegung beendet ist, erhält der Rufer der Methode ein Objekt der Klasse `DecomposedDocument` (vgl. Abschnitt 7.2.6.3, Seite 291) zurück, in dem sich die fertigen Wissensatome befinden.

Darüber hinaus bietet `SplittingAlgorithm` Methoden an, die es erlauben, Information über seinen aktuellen internen Zustand abzurufen (vgl. Abbildung 119).

Intern arbeitet `SplittingAlgorithm` mit einem Objekt, das das Interface `Decomposer` (vgl. Abschnitt 7.2.7.4, Seite 294) erfüllt. Die Klasse `MainWindow` sorgt dafür, dass für diesen Zweck immer ein Objekt übergeben wird, das die **Zerlegung des aktuell gewählten Dokuments** ausführen kann. Der genaue Vorgang zur Zerlegung eines Dokuments hängt wesentlich vom Dokument-Typ ab.

In diesem Zusammenhang vgl. auch die Ausführungen zu den Klassen `PlainTextSplitEditorKit` (Abschnitt 7.2.4.9, Seite 286), `HTMLSplitEditorKit` (Abschnitt 7.2.4.10, Seite 287), `HTMLPatternEditorKit` (Abschnitt 7.2.4.11, Seite 287) und `PdfEditorKit` (Abschnitt 7.2.8.3, Seite 299).

7.2.6.3 Klasse DecomposedDocument

`DecomposedDocument` kapselt sämtliche Information, die über ein bereits **zerlegtes elektronisches Dokument** von Bedeutung ist. Dabei bietet es die folgende Funktionalität an (vgl. Abbildung 119):

- Abfrage der URL des Original-Dokuments
- Zugriff auf alle aus dem Original-Dokument erzeugten Wissensatome

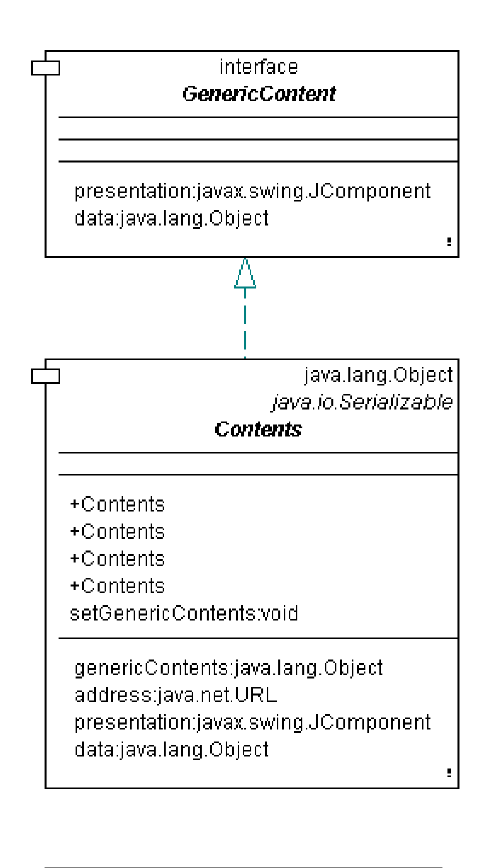
- Löschen von Wissensatomen, Einfügen von neuen Wissensatomen (aus der Zerteilung von zu großen Wissensatomen entstanden)

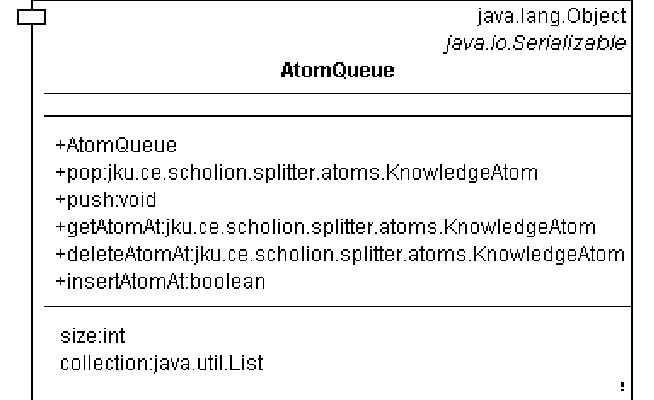
7.2.7 Package „atoms“

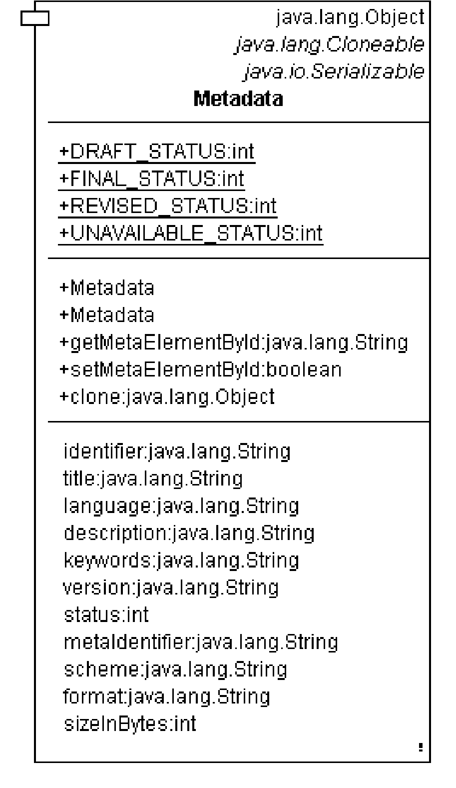
Das Package `splitter.atoms` beinhaltet alle Klassen, die für die **Modellierung von Wissensatomen** benötigt werden. Es definiert grundlegende Schnittstellen (Interfaces) und Klassen für die Zerlegung von Dokumenten und Speicherung von Wissensatomen. Zu diesem Zweck besteht das Package aus den folgenden, in den in Klammern genannten Abschnitten detailliert beschriebenen Teilen:

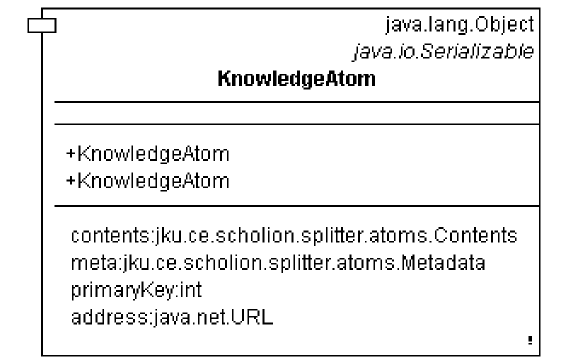
- Interface MetaExtractor (Abschnitt 7.2.7.1)
- Interface GenericContent (Abschnitt 7.2.7.2)
- Abstrakte Klasse Contents (Abschnitt 7.2.7.3)
- Interface Decomposer (Abschnitt 7.2.7.4)
- Klasse KnowledgeAtom (Abschnitt 7.2.7.5)
- Klasse Metadata (Abschnitt 7.2.7.6)
- Klasse AtomQueue (Abschnitt 7.2.7.7)
- Sub-Package „content“ (Abschnitt 7.2.7.8)

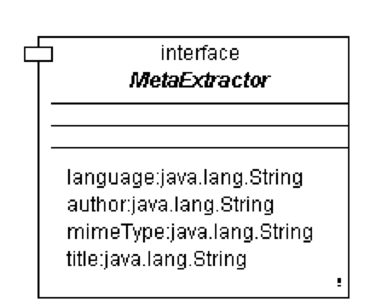
Abbildung 120 gibt einen Überblick über die einleitend genannten Teile des Packages `splitter.atoms`.

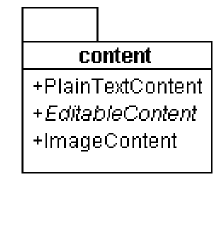












interface
Decomposer

+WORD_PATTERN:java.lang.String
+SENTENCE_PATTERN:java.lang.String
+PARAGRAPH_PATTERN:java.lang.String
+NEWLINE:java.lang.String

+*hasMoreSplices:boolean*
+*reset:void*
+*getOriginalAddress:java.net.URL*

nextSplice:java.lang.Object
documentMimeType:java.lang.String
contentLength:int
granularity:jku.ce.scholion.splitter.config.Granularity

Abbildung 120: Klassenmodell – Package splitter.atoms

7.2.7.1 Interface MetaExtractor

`MetaExtractor` definiert die Schnittstelle von, d.h. das Verhalten für alle Klassen, die aus einem elektronischen Dokument die für Dokument-Splitter relevanten Metadaten extrahieren (bzw. lesen) können. Es definiert `get()`- und `set()`-Methoden für die folgenden Aspekte eines Dokuments (vgl. Abbildung 120):

- Verfasser (Autor)
- Sprache
- Titel des Dokuments
- MIME-Type des elektronischen Original-Dokuments

7.2.7.2 Interface GenericContent

`GenericContent` definiert (auf einem sehr niedrigen Abstraktionsniveau) den kleinsten gemeinsamen Nenner aller Klassen, die **Inhalte eines Wissensatoms** modellieren. Jede Klasse, die das Interface `GenericContent` erfüllen will, muss folgende Funktionalität bereit stellen (vgl. Abbildung 120):

- Eine Präsentation von sich selbst erzeugen – der Begriff „Präsentation" meint dabei eine eigenständige GUI-Komponente, die leicht in eine andere Komponente der grafischen Benutzungsschnittstelle eingebaut werden kann
- Zugriff auf die Original-Daten bieten, wobei diese in einem beliebigen Objekt gekapselt sein dürfen

7.2.7.3 Abstrakte Klasse Contents

Die abstrakte Klasse `Contents` definiert die **Implementierung aller Methoden**, die den Klassen zur Modellierung von Inhalten eines Wissensatoms **gemeinsam** sind. `Contents` ist die Grundlage (d.h. die Superklasse) für alle Klassen im Sub-Package `content` (vgl. Abschnitt 7.2.7.8, Seite 296), die für die Kapselung der Inhalte von Wissensatomen in Dokument-Splitter eingesetzt werden.

Zusätzlich zu den bereits durch das Interface `GenericContent` (vgl. Abschnitt 7.2.7.2 weiter oben) definiert `Contents` eine `get()`- und `set()`-Methode für die Adresse (URL) des dem Wissensatom zu Grunde liegenden Original-Dokuments (vgl. Abbildung 120).

7.2.7.4 Interface Decomposer

`Decomposer` definiert die Schnittstelle von, d.h. das Verhalten für alle Klassen, die aus einem elektronischen Dokument für Dokument-Splitter **Wissensatome erzeugen** können. Jede Klasse, die das Interface `Decomposer` erfüllen will, muss folgende Funktionalität bereit stellen (vgl. Abbildung 120):

- Information über den internen Zustand erteilen (d.h. zur definierten Granularität der Wissensatome, modelliert durch eine Instanz von `Granularity`; sowie ein boolescher Wert, der angibt, ob weitere Inhalts-„Stückchen" zur Verfügung stehen)

- Die Möglichkeit, den internen Zustand des `Decomposer`-Objekts auf den Anfangszustand zurück zu setzen
- Zugriff auf die ursprüngliche Adresse (URL) des Original-Dokuments gewähren
- Den Inhalt, zertümmert in „Stückchen", die der definierten Granularität entsprechen, ausgeben
- Information über Länge und Datei-Typ (MIME-Type) des Original-Dokuments bereit stellen

7.2.7.5 Klasse KnowledgeAtom

`KnowledgeAtom` kapselt alle Daten, die zu einem **Wissensatom** von Bedeutung sind. Es enthält die folgenden Daten, zu denen jeweils eine `get()`- und eine `set()`-Methode zur Verfügung stehen (vgl. Abbildung 120):

- Eine fortlaufende eindeutige Nummer, die als Primärschlüssel für Wissensatome verwendet werden kann
- Den Inhalt als Instanz von `Contents` bzw. einer Subklasse davon (vgl. Abschnitt 7.2.7.3, Seite 294)
- Die Metadaten als Instanz von `Metadata` (vgl. Abschnitt 7.2.7.6 weiter unten)
- Seine ursprüngliche Adresse, das ist die URL des Original-Dokuments, aus dem die Instanz von `KnowledgeAtom` erzeugt wurde

7.2.7.6 Klasse Metadata

`Metadata` stellt die Schnittstelle für die **Modellierung der Metadaten** zu einem Wissensatom zur Verfügung. Die Schnittstelle von `Metadata` ist so entworfen, dass sämtliche, im LOM Standard (vgl. Abschnitt 4.3.2 IEEE Learning Object Metadata Standard, Seite 54 ff.) abgebildet werden können. Für die wichtigsten der im Standard definierten Metadaten werden eigene `get()`- und `set()`-Methoden implementiert (vgl. Abbildung 120).

7.2.7.7 Klasse AtomQueue

`AtomQueue` dient zur **Verwaltung** einer **Menge von Wissensatomen**, die aus einem elektronischen Dokument erzeugt wurden. Wie bereits aus dem Namen hervorgeht, ist `AtomQueue` als **Warteschlange** implementiert, bietet also Möglichkeiten zur dynamischen Manipulation der gegenwärtig gegenwärtig von ihr gespeicherten Menge der Wissensatome.

Folgende Operationen stehen in der Klasse `AtomQueue` zur Verfügung (vgl. Abbildung 120):

- Einfügen von Wissensatomen (Instanzen von `KnowledgeAtom`) am Ende der Warteschlange
- Entnehmen von Wissensatomen am Kopf der Warteschlange

- Zugriff auf Wissensatome durch Angabe ihrer Position (numerischer Wert) innerhalb der Warteschlange
- Einfügen von Wissensatomen innerhalb der Warteschlange (ermöglicht das „Vordrängen" von einzelnen `KnowledgeAtom`-Instanzen)
- Löschen von Wissensatomen, ohne sie zu entnehmen
- Abfragen der Anzahl von gegenwärtig in der Warteschlange gespeicherten Wissensatomen

7.2.7.8 Sub-Package „content"

Das Sub-Package `content` im Package `splitter.atoms` gruppiert alle Klassen, die das Interface `GenericContent` (vgl. Abschnitt 7.2.7.2, Seite 294) erfüllen und als Subklassen von `Contents` (vgl. Abschnitt 7.2.7.3, Seite 294) implementiert sind. Mit anderen Worten, im Sub-Package `content` ist die gesamte **Anwendungslogik zur Repräsentation** aller vom Dokument-Splitter unterstützten **Inhalte** enthalten. Es besteht aus folgenden Teilen:

- Interface EditableContent (Punkt 7.2.7.8.1)
- Klasse PlainTextContent (Punkt 7.2.7.8.2)
- Klasse ImageContent (Punkt 7.2.7.8.3)

Durch Hinzufügen von weiteren Klassen kann die Funktionalität des Dokument-Splitters noch erweitert werden. So wäre es denkbar, z.B. eine Klasse `AudioContents` zu implementieren. Mit anderen Worten, die Architektur des Dokument-Splitters ist besonders in Bezug auf die Speicherung der Inhalte offen und erweiterbar gestaltet. Abbildung 121 gibt einen Überblick über die oben genannten Teile des Packages `splitter.atoms.content`.

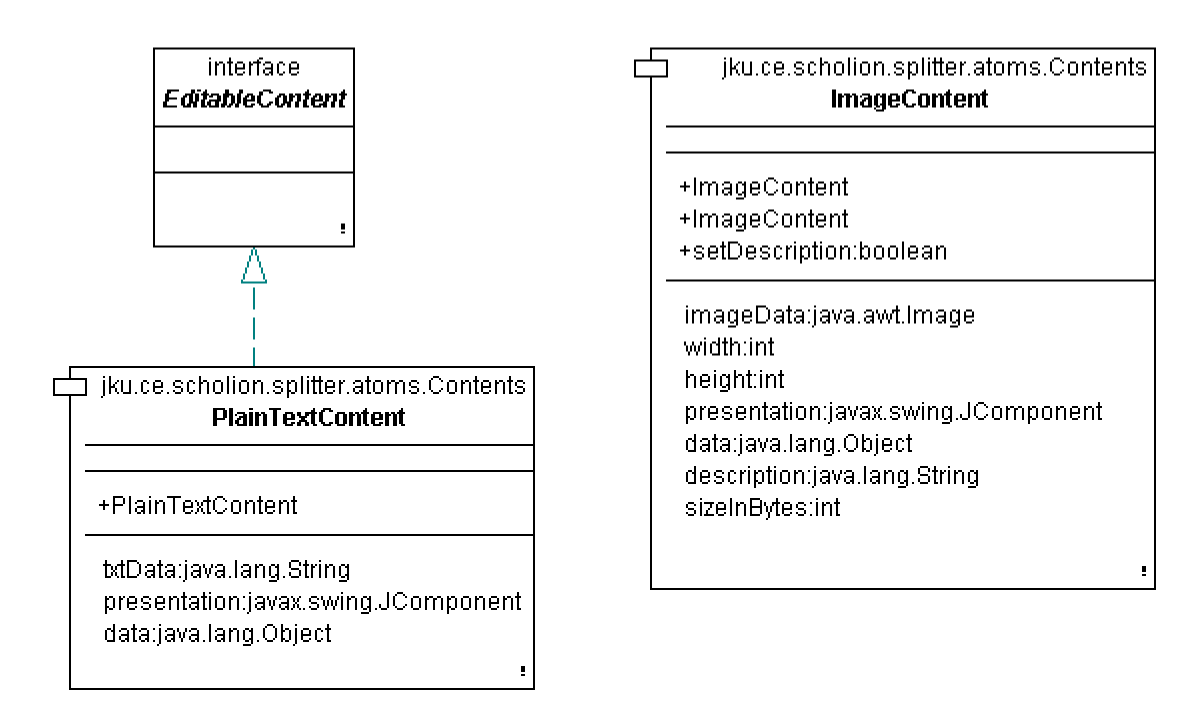


Abbildung 121: Klassenmodell – Package splitter.atoms.content

7.2.7.8.1 Interface EditableContent

Das Interface `EditableContent` ist ein leeres Java-Interface (so genanntes Marker-Interface). Es dient lediglich dazu, jene Subklassen von `Contents` zu kennzeichnen, die vom Benutzer nach Erstellung eines Wissensatoms noch editiert werden können bzw. dürfen.

7.2.7.8.2 Klasse PlainTextContent

`PlainTextContent` modelliert Inhalte eines Wissensatoms, die aus **reinem Text** bestehen. Der Inhalt einer Instanz von `PlainTextContent` wird in einem Objekt der Klasse `java.lang.String` gespeichert. Die Klasse `PlainTextContent` stellt die in den Interfaces `GenericContent` und `EditableContent` vorgesehenen Zugriffsmethoden zur Verfügung (vgl. Abbildung 121).

7.2.7.8.3 Klasse ImageContent

`ImageContent` modelliert Inhalte eines Wissensatoms, die aus **Bild-Daten** bestehen. Der Inhalt einer Instanz von `ImageContent` wird in einem Objekt der Klasse `java.awt.Image` gespeichert. Die Klasse `ImageContent` stellt die im Interface `GenericContent` vorgesehenen Zugriffsmethoden zur Verfügung (vgl. Abbildung 121).

7.2.8 Package „pdf"

Im Package `splitter.pdf` ist die gesamte **Anwendungslogik** für das Öffnen, Lesen (Parsen) und Zerlegen von elektronischen Dokumenten im **Portable Document Format** (PDF) zusammen gefasst. Die genannten Aufgaben sind durchaus nicht trivial. (Im Ab-

schnitt 9.2 Future Work, Seite 319 f. werden die Probleme beschrieben, die beim Öffnen, Lesen und Zerlegen von PDF-Dokumenten auftreten.)

Zum Package `splitter.pdf` gehören auch Klassen, die an der grafischen Benutzungsschnittstelle des Dokument-Splitters eine **adäquate** (d.h. möglichst zur Original-Version des PDF-Dokuments identische) **Darstellung** eines PDF-Dokuments erzeugen können.

Das Package besteht aus den folgenden, in den in Klammer angegebenen Abschnitten detailliert beschriebenen Teilen:

- Klasse JPdfPane (Abschnitt 7.2.8.1)
- Klasse PdfDocument (Abschnitt 7.2.8.2)
- Klasse PdfEditorKit (Abschnitt 7.2.8.3)
- Sub-Package „importing“ (Abschnitt 7.2.8.4)
- Sub-Package „readers“ (Abschnitt 7.2.8.5)

Abbildung 122 gibt einen Überblick über die oben genannten Teile des Packages `splitter.pdf`.

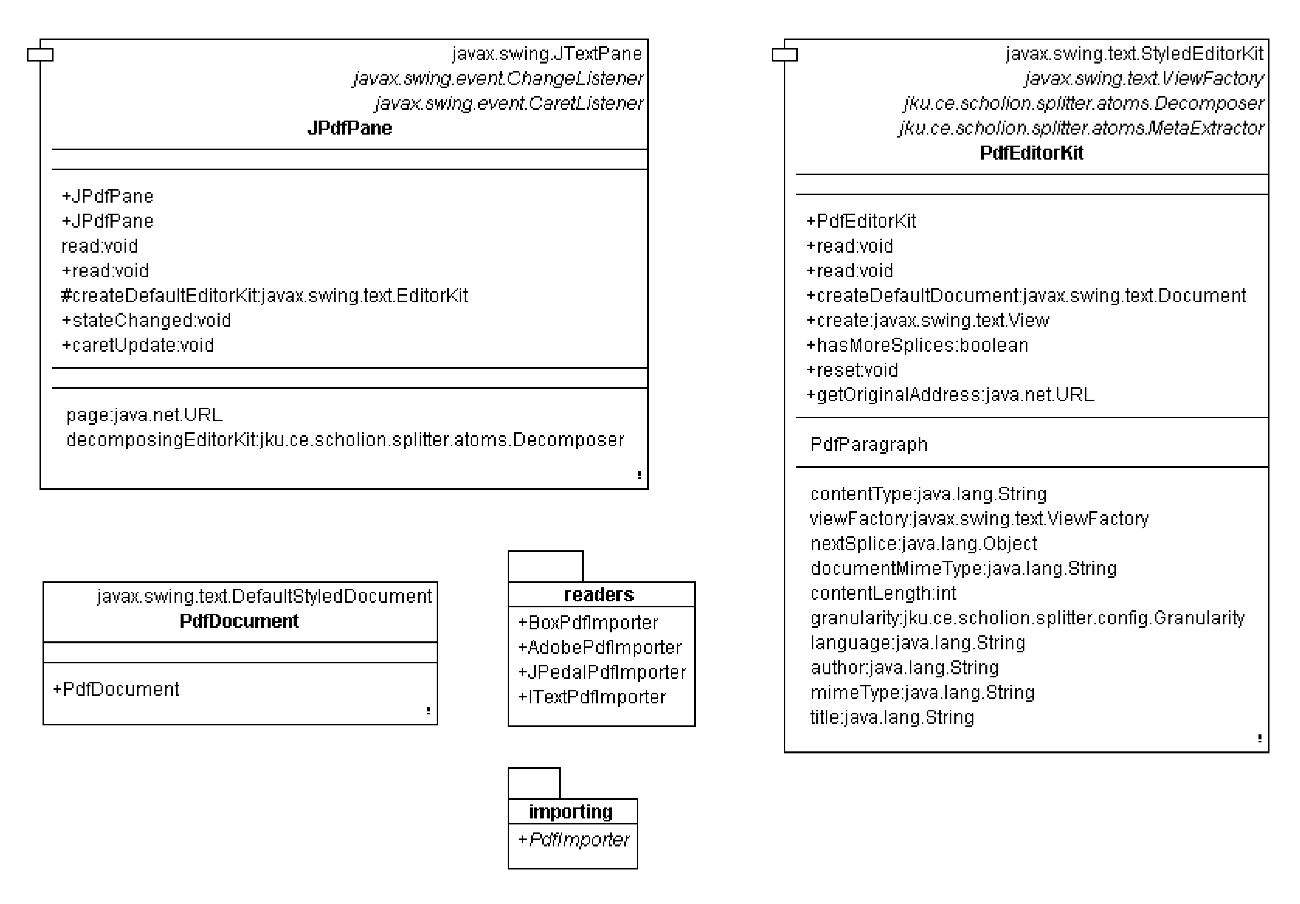


Abbildung 122: Klassenmodell – Package splitters.pdf

7.2.8.1 Klasse JPdfPane

`JPdfPane` dient zur **Darstellung von PDF-Dokumenten** an der grafischen Benutzungsschnittstelle (GUI) des Dokument-Splitters. Die Klasse ist dabei zuständig für das Laden (Parsen) eines PDF-Dokuments und das Anzeigen des Dokument-Inhalts an der GUI.

Unterstützung bei dieser Aufgabe erhält das `JPdfPane` durch die Verwendung einer Klasse, die das Interface `PdfImporter` erfüllt (vgl. Abschnitt 7.2.8.4, Seite 299). In der gegenwärtigen Version des Dokument-Splitters wird für diesen Zweck die Klasse `BoxPdfImporter` verwendet (vgl. Punkt 7.2.8.5.1, Seite 302).

Zur Anzeige der Inhalte aus einem PDF-Dokument verwendet `JPdfPane` seinerseits eine Instanz von `PdfDocument` (vgl. Abschnitt 7.2.8.2 weiter unten). `JPdfPane` ist auf oberster Ebene in eine Instanz von `MainWindow` (vgl. Punkt 7.2.1.1.1, Seite 276) eingebettet.

7.2.8.2 Klasse PdfDocument

`PdfDocument` ist spezialisiert darauf, die **Inhalte** aus einem PDF-Dokument in **eigenen Datenstrukturen** zu speichern, so dass die Anzeige des Dokuments an der grafischen Benutzungsschnittstelle so weit wie möglich erleichtert wird.

Die Funktionalität von `PdfDocument` ist dabei identisch zu jener seiner Superklasse (`javax.swing.text.DefaultStyledDocument`, vgl. Abbildung 122).

7.2.8.3 Klasse PdfEditorKit

`PdfEditorKit` stellt Funktionalität zur Verfügung, um in die **Darstellung eines PDF-Dokuments** (mit Hilfe einer Instanz von `JPdfPane`, vgl. Abschnitt 7.2.8.1 weiter oben) zusätzlichen Text einzufügen, bestehenden Text zu bearbeiten oder zu löschen. Wichtig ist die Feststellung, dass diese Änderungen nur die Darstellung, nicht aber das Original-Dokument selbst verändern. Die Berechnung der Wissensatome erfolgt dann aufgrund der **lokal veränderten** Darstellung.

Dieses Konzept ermöglicht es den Benutzern, bereits vor Beginn der (rechenintensiven) Zerlegung jene Teile aus dem Dokument zu löschen, die sie für unnötig erachten. Auf diese Weise kann bei der Zerlegung die Rechenzeit verringert werden.

`PdfEditorKit` erfüllt die Interfaces `Decomposer` (vgl. Abschnitt 7.2.7.4, Seite 294) und `MetaExtractor` (vgl. Abschnitt 7.2.7.1, Seite 294), ist also ein wichtiges Hilfsmittel bei der Zerlegung der im Dokument-Splitter geöffneten PDF-Dokumente (siehe Abbildung 122).

7.2.8.4 Sub-Package „importing“

Das Sub-Package `importing` im Package `splitter.pdf` modelliert alle grundlegenden Aspekte, die mit den technischen Problemen beim Lesen und Parsen (als Importieren bezeichnet) von PDF-Dokumenten auftreten. Gegenwärtig besteht das Package nur aus einem einzigen Interface:

- Interface PdfImporter (Punkt 7.2.8.4.1)

Zweck des Packages `importing` ist es, die **Grundlagen** für die im Package `readers` (vgl. Abschnitt 7.2.8.5, Seite 301) implementierten Klassen **zusammen zu fassen**. Ursprünglich war zusätzlich geplant, in einer abstrakten Basisklasse die Gemeinsamkeiten aller Klassen, die sich mit dem Import von PDF-Daten befassen, zusammen zu fassen. Aufgrund der großen Unterschiede zwischen den dazu verwendeten technischen Konzepten (von welchen der Leser sich in den Punkten 7.2.8.5.1 bis 7.2.8.5.4, Seite 302 ff. ein Bild zeichnen kann) hat sich dieses Vorhaben aber schließlich als unrealistisch heraus kristallisiert.

Abbildung 123 zeigt das Klassenmodell des oben beschriebenen Packages `splitter.pdf.importing`.

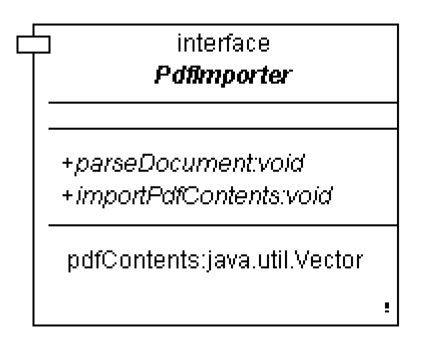


Abbildung 123: Klassenmodell – Package splitter.pdf.importing

7.2.8.4.1 Interface PdfImporter

`PdfImporter` definiert die Schnittstelle für alle Klassen, die zum Import von Daten im PDF-Format eingesetzt werden. (**Hinweis**: alle diese Klassen sind im Package `splitter.pdf.readers` zusammen gefasst, vgl. Abschnitt 7.2.8.5 weiter unten.) Jede Klasse, die das Interface `PdfImporter` erfüllen will, muss folgende Funktionalität zur Verfügung stellen (vgl. Abbildung 123):

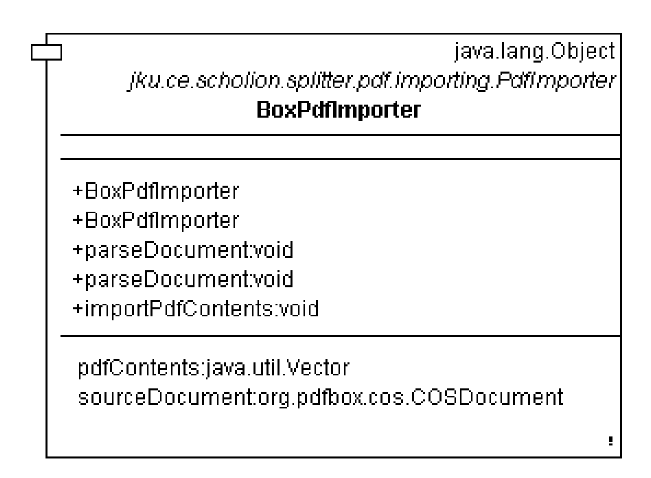
- **Parsen** eines PDF-Dokuments: dabei wird der Inhalt aus einer Datei im PDF-Format aus einem Eingabe-Zeichenstrom gelesen und entsprechend der PDF-Spezifikation [PdfSpec, 2001] interpretiert. Aus der gelesenen Information kann eine interne Repräsentation der Daten gebildet werden.
- **Importieren** der gelesenen Inhalte aus dem PDF-Dokument in eine Instanz von `PdfDocument` innerhalb einer Instanz von `JPdfPane` (vgl. Abschnitt 7.2.8.1, Seite 299).
- **Zugriff auf die Inhalte** des PDF-Dokuments in einer Form, die von der Implementierung der das Interface `PdfImporter` erfüllenden Klasse bestimmt wird. In der Regel wird der Inhalt, in Absätze zerteilt, innerhalb eines Containers an den Rufer zurück gegeben werden.

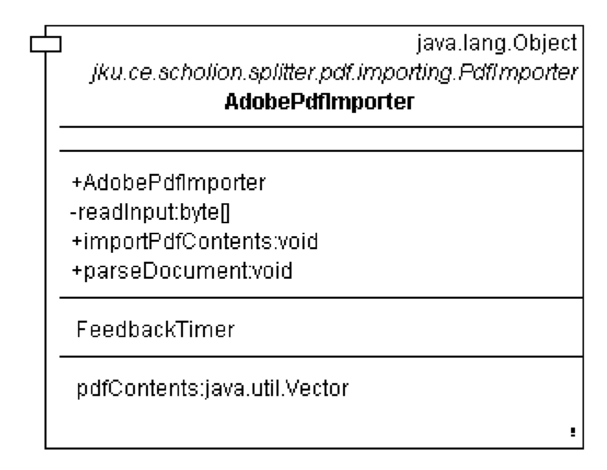
7.2.8.5 Sub-Package „readers“

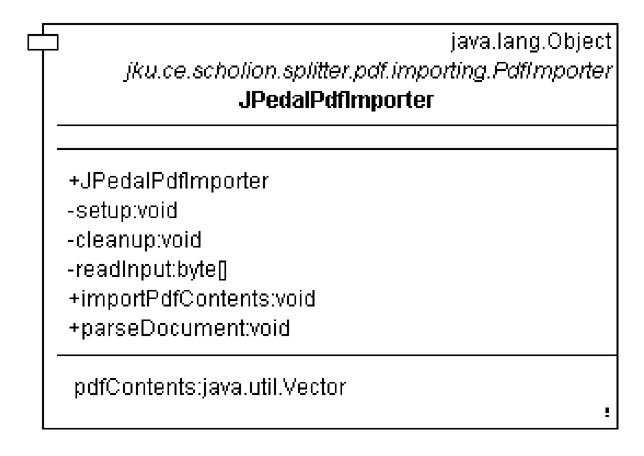
Das Sub-Package `readers` im Package `splitter.pdf` enthält alle Klassen, die das Interface `PdfImporter` (vgl. Punkt 7.2.8.4.1 weiter oben) erfüllen. Hinter der Implementierung in den Klassen stehen meist völlig verschiedene technische Konzepte (wie auf den kommenden Seiten näher erläutert werden wird). Das Package besteht aus den folgenden Klassen:

- Klasse BoxPdfImporter (Punkt 7.2.8.5.1)
- Klasse AdobePdfImporter (Punkt 7.2.8.5.2)
- Klasse JPedalPdfImporter (Punkt 7.2.8.5.3)
- Klasse ITextPdfImporter (Punkt 7.2.8.5.4)

Abbildung 124 gibt einen Überblick über die oben genannten Teile des Packages `splitter.pdf.readers`.







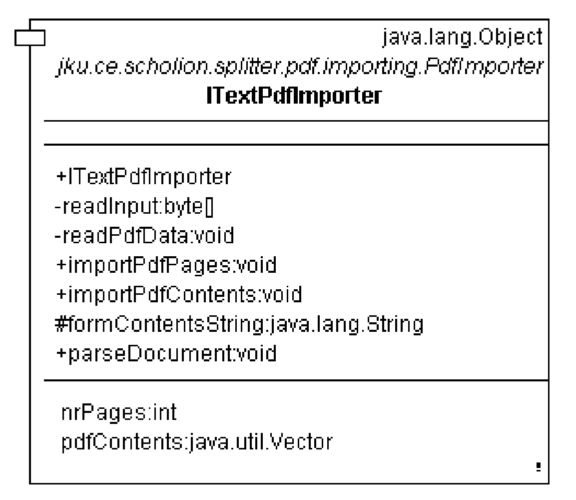


Abbildung 124: Klassenmodell – Package „splitter.pdf.readers“

Hinweis: Alle in der Abbildung 124 gezeigten Klassen erfüllen, wie bereits erwähnt, das Interface `PdfImporter`. Folglich ist die Funktionalität aller auf den kommenden Seiten

beschriebenen Klassen aus dem Package `splitter.pdf.readers` identisch. Aus diesem Grund verzichte ich darauf, bei der Beschreibung der Klassen den Hinweis auf deren Funktionalität jedes Mal zu erwähnen. Statt dessen konzentrieren sich die Ausführungen der kommenden Seiten auf die Aspekte der Realisierung und technischen Grundlagen dieser Funktionalität.

7.2.8.5.1 Klasse BoxPdfImporter

`BoxPdfImporter` verwendet die **Java-Klassenbibliothek „PdfBox"** [PdfBox, 2002]. PdfBox erlaubt einen Zugriff auf die Inhalte einer PDF-Datei auf hohem Abstraktionsniveau. Objekte aus einer PDF-Datei werden in Java-Objekte abgebildet. Ein Zugriff auf die einzelnen Objekte ist allerdings nur dann notwendig, wenn bestimmte, für spezielle Anforderungen notwendige Veränderungen an den Algorithmen vorgenommen werden müssen. Ansonsten liefert die Schnittstelle, mit der man auf ein Dokument als Ganzes zugreifen kann, brauchbare Ergebnisse, die vom Dokument-Splitter hervorragend verarbeitet werden können.

Für Details zur Java-Klassenbibliothek PdfBox wird der Leser auf [PdfBox, 2002] verwiesen.

Vom Autor veränderte Algorithmen kommen nur in der Klasse `AdvancedPdfTextStripper` (vgl. Abschnitt 7.2.4.4, Seite 285) zum Einsatz. Für alle restlichen Anwendungen wurden die bestehenden Algorithmen verwendet.

7.2.8.5.2 Klasse AdobePdfImporter

`AdobePdfImporter` verwendet die Schnittstelle, wie sie von **Adobe Pdf Viewer**, einer Java-Klassenbibliothek, zur Verfügung gestellt wird [Adobe, 2002].

Unter Zuhilfenahme von Pdf Viewer ist der Zugriff auf die Inhalte einer PDF-Datei nur auf sehr geringem Abstraktionsniveau möglich. Die Schnittstelle der Adobe Klassenbibliothek liefert **einzelne PDF-Objekte** zusammen mit Daten zu deren Positionierung auf einer Seite (Koordinaten). All diese Daten müssen von einer Anwendung erst **interpretiert** werden und müssen in eine authentische (d.h. dem Original-Dokument entsprechende) Darstellung unter Verwendung eigener, zu entwerfender Algorithmen einfließen.

Aus diesem Grund bietet `AdobePdfImporter` einen Umfang an Funktionalität, der dem von `BoxPdfImporter` unterlegen ist. Deshalb wird `AdobePdfImporter` im Dokument-Splitter nicht für das Erzeugen von Wissensatomen verwendet.

Sehr mächtig ist hingegen die Schnittstelle des Adobe Pdf Viewer zur **Erzeugung von Grafiken** aus PDF-Dokumenten. Diese Funktionalität wird im `PdfPreviewFrame` genutzt und zur Generierung von Vorschau-Grafiken verwendet (vgl. Punkt 7.2.1.1.3, Seite 277).

7.2.8.5.3 Klasse JPedalPdfImporter

`JPedalPdfImporter` verwendet die **Java-Klassenbibliothek „JPedal"**. JPedal ist ein Akronym und steht für „Java PDF Extraction Decode Access Library" [JPedal, 2003]. Bei JPedal ist, wie auch bei Adobe Pdf Viewer, ein Zugriff auf die Inhalte einer PDF-Datei bedauerlicherweise nur auf niedrigem Abstraktionsniveau möglich.

Darüber hinaus scheint die Implementierung von JPedal in der gegenwärtigen Version noch mit **gröberen Fehlern** behaftet zu sein. Aus diesem Grund ist die Implementierung von `JPedalPdfImporter`, obwohl die Funktionalität wie im Interface `PdfImporter` vorgesehen erfüllt wird, als eine prototypische zu betrachten. `JPedalPdfImporter` kommt daher in der aktuellen Version des Dokument-Splitters nicht zum Einsatz.

7.2.8.5.4 Klasse ITextPdfImporter

`ITextPdfImporter` setzt auf der Schnittstelle der **Java-Klassenbibliothek iText** (vgl. [iText, 2003] und [Sullivan, 2003]) auf. Die Funktionalität von iText ist sehr mächtig. Der Zugriff auf ein PDF-Dokument kann auf hohem Abstraktionsniveau erfolgen, wobei ein Dokument Schnittstellen für den Umgang mit seinen Objekten anbietet (vgl. [iText, 2003]).

Der Fokus von iText liegt jedoch leider auf dem Erstellen von PDF-Dokumenten. Funktionalität zum Extrahieren von Inhalten aus einem Dokument im PDF-Format sind ist nur in Ansätzen vorhanden. Aus diesem Grund schafft `ITextPdfImporter` keine befriedigenden Ergebnisse beim Importieren von PDF-Dateien. Die Implementierung geht über ein experimentelles oder prototypisches Stadium nicht hinaus. Deshalb wird `ITextPdfImporter` in der aktuellen Version des Dokument-Splitters nicht eingesetzt.

7.3 Zusammenfassung

Im Kapitel 7 wurde gezeigt, wie das im Kapitel 6 entwickelte Contentpool-Konzept in eine Implementierung umgesetzt wurde. Die hier beschriebenen Werkzeuge stellen ein Beispiel dar, wie das Contentpool-Konzept realisiert werden kann.

Mit dem Abschluss der Implementierung wurden folgende Ziele der Diplomarbeit erreicht:

- Ein Werkzeug zur Zerlegung von Dokumenten wurde implementiert.
- Das Datenmodell des Contentpools erfüllt die Kriterien Interoperabilität und Offenheit. Speicherbedarf und Zugriffszeit konnten sehr niedrig gehalten werden.

Folgendes Ziel wurde lediglich teilweise erreicht:

- Werkzeuge zur Suche in und Darstellung von Wissensatomen wurden erfolgreich implementiert. Die Implementierung eines Werkzeugs zur Erstellung von Kursmaterialien und Nachschlagewerken konnte aus Zeitgründen nicht realisiert werden.

Folgende Fragestellungen der Diplomarbeit wurden beantwortet:

- Die Zerlegung von Dokumenten wurde in einem Werkzeug implementiert und erfolgreich getestet.
- Für die Speicherung von Wissensatomen und semantischen Beziehungen konnte ein Datenmodell spezifiziert werden, das alle zuvor definierten Ziele erfüllt.
- Die Darstellung von Wissensatomen wurde in einer grafischen Benutzungsschnittstelle realisiert.

Folgende Fragestellung der Diplomarbeit blieb unbeantwortet:

- Obwohl auf konzeptueller Ebene die Anforderungen für die Berücksichtigung semantischer Zusammenhänge bei Zerlegung von Dokumenten bestimmt werden konnten, ist es nicht gelungen, die Anforderungen in das Werkzeug zur Zerlegung von Dokumenten zu implementieren. Der zeitliche Aufwand für die Umsetzung der Anforderungen wurde unterschätzt.

Bei der Implementierung der Werkzeuge wurden die Ziele für Entwurf und Implementierung (vgl. Abschnitt 2.3, Seite 18 ff.) überwiegend erfüllt:

- Allgemeine Benutzbarkeitskriterien wurden bei der Gestaltung von Benutzungsschnittstellen als oberstes Prinzip beachtet.
- Mit dem Werkzeug zur Zerlegung von Dokumenten ist die Aufbereitung von Wissensatomen möglich. Nicht realisiert wurde, wie erwähnt, die Funktionalität zur Erstellung von semantischen Zusammenhängen. Es ist jedoch möglich, die semantischen Zusammenhänge im Contentpool (Werkzeug „Topic Editor") manuell zu editieren.
- Die Speicherung von Wissensatomen und semantischen Zusammenhängen wurde ermöglicht.
- Die Implementierung erfüllt die sonstigen Designziele.

8 Test der Software

Im Rahmen eines **Projektstudiums** am Institut für Wirtschaftsinformatik, Schwerpunkt Communications Engineering, wurden die Tools zur Verwaltung des Contentpools („Topic Map Tools" [Wizany, 2003]) einem intensiven Test unterzogen. Ziel der Tests war es, Mängel bei der **Umsetzung der Anforderungen** festzustellen und zu korrigieren. [Wizany, 2003] Dieses Kapitel beschäftigt sich mit den Ergebnissen der Tests.

Das Werkzeug zur Zerlegung von Dokumenten, der Dokument-Splitter, konnte aus Mangel an Zeit und personellen Ressourcen noch keinem Test durch die Benutzer unterzogen werden. Hier wurden lediglich White Box Tests durch den Autor dieser Diplomarbeit während der Implementierung vorgenommen.

Zu Beginn dieses Kapitels wird die bei den Tests verwendete Testmethodik vorgestellt. Anschließend werden, nach den einzelnen Werkzeugen gemäß der Architektur in Abbildung 52 (Architektur – Scholion WB+ Contentpool-Verwaltung; siehe Seite 153) unterteilt, die Ergebnisse der Tests präsentiert. Das Kapitel 8 enthält gemäß dieser Einteilung die folgenden Abschnitte:

Hinweis: Für das Verständnis der in den Abschnitten 8.2, 8.3, 8.4 und 8.5 gezeigten Testergebnisse ist es hilfreich, die Benutzerdokumentation für die Werkzeuge der Contentpool Verwaltung zu konsultieren [Berger, 2003]. In der Benutzerdokumentation sind unter Anderem zahlreiche Abbildungen der Benutzungsschnittstelle enthalten, auf die sich die im Kapitel 8 präsentierten Testergebnisse beziehen.

8.1 Methodik für die Tests der Programme

Zum Testen der im Rahmen des Projekts Scholion WB+ implementierten Software wurden folgende **Testmethoden** verwendet:

- White-Box Tests (siehe Abschnitt 8.1.1)
- Black-Box Tests (siehe Abschnitt 8.1.2)

Beide genannten Methoden sind Testmethoden des Software Engineering, die zur **dynamischen Analyse** von Softwareprodukten im Rahmen **analytischer Qualitätssicherungs-Maßnahmen** dienen (vgl. C. Floyd und H. Züllighoven in [RechPom, 1999], S. 783). Ziel beider Testmethoden ist es, Fehler im Programm zu finden, nicht aber, dessen Korrektheit (Fehlerfreiheit) zu beweisen (vgl. ebd.).

Den Abschluss des Abschnitts 8.1 bildet ein Abriss über die **formale Klassifikation** von mit Hilfe der Testmethoden gefundenen Fehlern (siehe Klassifikation von Fehlern, Ab-

schnitt 8.1.3). Die Erläuterung der Klassifikation ist für die später im vorliegenden Kapitel präsentierten Testergebnisse von Bedeutung.

In den folgenden Abschnitten bringt der Autor ergänzende Erläuterungen zu den oben genannten Testmethoden und beschreibt die angewandte Vorgehensweise beim Testen mit Hilfe der Testmethoden.

8.1.1 White-Box Tests

Die White-Box Tests erfolgten bereits **während der Implementierung** der Werkzeuge als Maßnahme der **ständigen Qualitätssicherung** des programmierten Codes (so genannte „analytische Qualitätssicherung" [RechPom, 1999], S. 783).

Beim White-Box Test oder auch „Strukturtest" (Heinrich [HeinRoith, 1998], S. 509) wird die **innere Struktur** eines Testobjekts (im vorliegenden Fall eine Komponente in einem Programm für Scholion WB+) überprüft. Dazu werden aus der Struktur des Projekts (hier Scholion WB+) Testdaten abgeleitet, mit deren Hilfe getestet wird, ob sich das Testobjekt **konform zu seiner Spezifikation** verhält. Jeder mögliche Kontrollpfad durch das Testobjekt ist dabei Gegenstand des Tests (vgl. [HeinRoith, 1998], S. 509).

White-Box Tests wurden während des Projekts Scholion WB+ regelmäßig eingesetzt, um unmittelbar nach der Programmierung neuer Komponenten (Klassen oder Methoden) deren Qualität sicher zu stellen. Vom Autor wurde umgehend nach Fertigstellung einer Komponente überprüft, ob mit dem neu implementierten Code die beabsichtigte Funktionalität realisiert werden konnte. Zu diesem Zweck wurden folgende Hilfsmittel verwendet:

- Testpläne und Testtreiber
- Testklassen auf der Basis von JUnit (so genannte Unit Tests) [JUnit, 2002]

8.1.2 Black-Box Tests

Mit Black-Box Tests zu den Werkzeugen der Contentpool-Verwaltung wurde begonnen, so bald eine **erste lauffähige Version** zur Verfügung stand. Auch Black-Box Tests sind eine Maßnahme der **analytischen Qualitätssicherung**, um die Qualität des programmierten Codes zu überprüfen und zu gewährleisten (vgl. C. Floyd und H. Züllighoven in [RechPom, 1999], S. 783).

Beim Black-Box Test oder auch „Funktionstest" (Heinrich [HeinRoith, 1998], S. 227) wird die **Funktion eines Testobjekts** (im vorliegenden Fall eine Komponente in einem Programm für Scholion WB+) überprüft. Die Testdaten für einen Black-Box Test werden aus der **Spezifikation des Testobjekts** abgeleitet. Auf die innere Struktur des Testobjekts wird nicht eingegangen (vgl. [HeinRoith, 1998], S. 227). Der Test überprüft, ob sich das Testobjekt so wie in der Spezifikation festgeschrieben verhält.

Der wichtigste Testlauf nach der Black-Box Testmethode war das Projektstudium „Test von Topic Map Tools" von B. Wizany [Wizany, 2003]. Während des Projektstudiums wurden die Werkzeuge der Contentpool-Verwaltung nach folgender Vorgehensweise getestet:

1. Für alle Komponenten aus Werkzeugen der Contentpool-Verwaltung in Scholion WB+ wurden gemäß der Spezifikation der Anforderungen **Testdaten** abgeleitet und Testpläne entworfen.
2. Die Testdaten und Testpläne dienten als Grundlage für den anschließend durchgeführten **Test der Werkzeuge**. Gefundene Fehler waren zu dokumentieren und dem Autor der Diplomarbeit zu kommunizieren.
3. Der **Code** der Komponenten wurde **überarbeitet** mit dem Ziel, alle der im zweiten Schritt gefundenen Fehler auszubessern.
4. Schließlich wurde gemäß des in Punkt 1 entworfenen Testplans und der in Punkt 1 abgeleiteten Testdaten der gesamte Ablauf des **Black-Box Tests** (gemäß des in Punkt 2 gewählten Ablaufs) **wiederholt**. Zweck dieses Vorgehens war es, die angebliche Korrektur der gefundenen Fehler zu verifizieren.

8.1.3 Klassifikation von Fehlern

Jedem während des Testens gefundenen Fehler wird, um diesen später eindeutig identifizieren zu können, eine Klassifikation zugewiesen. Die Klassifikation besteht aus folgenden Teilen (vgl. [Wizany, 2003], S. 2):

- **Art des Fehlers**, mit dessen Hilfe eine grobe Einordnung der Ursache des Fehlers getroffen werden kann. Durch die Art des Fehlers können zudem logische Fehler (d.h. Fehler im Programmablauf) von leichter zu behebenden Fehlern in der Benutzungsschnittstelle (z.B. reinen Tippfehlern) unterschieden werden. Mögliche Ausprägungen für die Art des Fehlers sind:
    - TF … „Tippfehler"
    - LF … „Layout-Fehler"
    - PF … „Fehler im Programmablauf"
    - UF … „Fehler oder Mangel bezüglich der Usability"
- **Name des Programmes**, in dem der Fehler gefunden wurde. An Stelle des vollen Namens wird eine **Abkürzung**, die aus zwei Buchstaben besteht, verwendet. Für das Werkzeug „Content-Navigator" wird die Abkürzung „CN", für das Werkzeug „Topic-Editor" die Abkürzung „TE" und für das Werkzeug „Topicmap-Manager" die Abkürzung „TM" definiert. Lässt sich ein Fehler nicht genau einem Programm zuordnen, so wird die Abkürzung „ALLG" (allgemeiner Fehler) vergeben.
- Innerhalb der Fehler-Kategorie, die sich aus der Kombination der beiden oben genannten Teile ergibt, wird schließlich eine **fortlaufende Nummer** vergeben.

Die **Syntax** der Klassifikation eines Fehlers ist wie folgt definiert. Je eine Ausprägung der drei oben genannten Teile der Klassifikation ergibt einen Klassifikations-Schlüssel nach der Anordnungsvorschrift:

```
<Art des Fehlers> "." <Programm> "." <fortlaufende Nummer>
```

Nach der obigen Definition wäre zum Beispiel die Zeichenkette „PF.TM.14" eine gültige Klassifikation. Sie bezeichnet den vierzehnten Fehler innerhalb der Fehlerkategorie „Programmablauf-Fehler im Topicmap-Manager".

Der Rest des achten Kapitels beschäftigt sich mit den Ergebnissen der von Wizany durchgeführten Black-Box Tests (siehe [Wizany, 2003]). Die vollständige Fehlerfreiheit und Korrektheit der Werkzeuge zur Verwaltung des Contentpools ist damit, wie bereits weiter oben erwähnt, nicht bewiesen. Auf Grundlage der Testergebnisse konnte lediglich die Zahl der im Programm enthaltenen Fehler reduziert werden.

8.2 Testergebnisse: Content-Navigator

Auf den kommenden Seiten werden die Fehler beschrieben, die durch den Black-Box Test der Contentpool-Verwaltung im **Werkzeug „Content-Navigator"** (vgl. Abbildung 52: Architektur – Scholion WB+ Contentpool-Verwaltung, Seite 153, im linken oberen Teil der Abb.) aufgezeigt werden konnten. Die folgende Tabelle 11 gibt außerdem Auskunft darüber, ob, und wenn ja, wie ein Fehler bereits behoben werden konnte.

Die hier vorgestellten Ergebnisse des Black-Box Tests wurden aus [Wizany, 2003] (S. 3 – 28) entnommen. Bei einzelnen Fehlern kann es sein, dass Maßnahmen zur Behebung und Status des Fehlers nicht mit dem Quelldokument überein stimmen. In solchen Fällen wurde nach Fertigstellung des Dokuments von B. Wizany noch eine Fehlerkorrektur durchgeführt.

Tabelle 11: Fehler im Content-Navigator

Klassifikation	Kurze Beschreibung	Maßnahme(n) zur Behebung	Status
TF.CN.01	Tippfehler im Menü.	Der Tippfehler wurde korrigiert.	behoben
LF.CN.01	Navigationsleiste (History der besuchten Topics) ist ab einer Länge von 4 bis 5 Einträgen nicht mehr lesbar.	Das erste und die drei zuletzt besuchten Topics bleiben sichtbar. Alle dazwischen liegenden Topics werden nicht dargestellt, sind aber in der History gespeichert.	behoben
LF.CN.02	Bei sehr langen Namen von Topics kommt es zu Überschneidungen mit Namen der benachbart angezeigten Topics.	Sehr lange Namen werden verkürzt dargestellt.	behoben
LF.CN.03	Zeigt man mit der Maus auf ein Topic, wird der dann fett dargestellte Name am Rand des Inhalts-Frames abgeschnitten.	Sehr lange Namen werden verkürzt dargestellt. Würde ein Name trotzdem noch über den Rand eines Frames hinaus reichen, wird der Name so verschoben, dass er sichtbar bleibt.	behoben
LF.CN.04	Occurrences eines Topics werden so dargestellt, dass sie sich mit den Topics überschneiden.	Dieser Fehler trat vorübergehend auf. Der Frame, in dem Occurrences angezeigt werden, wurde wieder an die richtige Stelle platziert.	behoben
LF.CN.05	Bei Auswahl von „Design 2" wird keine Navigationsleiste (History) mehr ange-	Das Design wurde korrigiert.	behoben

	zeigt.		
PF.CN.01	Bei sehr „großen"[37] Topic Maps beträgt die Berechnungszeit für die Baumdarstellung der Topic-Liste mehrere Minuten.	Der Baum der in der Topic Map enthaltenen Topics wird nun nicht mehr vollständig berechnet. Statt dessen berechnet der Content-Navigator ausschließlich den vom Benutzer angeforderten Teilbaum.	behoben
PF.CN.02	Zwei miteinander verknüpfte Topics werden angezeigt, die Association zwischen den Topics jedoch nicht.	Die Ursache des Fehlers scheinen ungültige Zeichen in importierten Topic Maps zu sein. Die ungültigen Zeichen können vom Topic Map Parser jedoch nicht erkannt werden.	offen
PF.CN.03	Beim Exportieren einer Topic Map wird die zusätzliche Datei-Erweiterung „.xml" angehängt.	Die Datei-Erweiterung „.xtm" muss im System des Benutzers registriert werden.	behoben
PF.CN.04	Der Export einer Topic Map öffnet ein neues Fenster, das anschließend geöffnet bleibt.	Beim Export einer Topic Map öffnet sich kein neues Fenster mehr.	behoben
PF.CN.05	Bei Auswahl eines Association Type, Member Type oder Occurrence Type werden keine verknüpften Topics angezeigt.	Dies liegt im Konzept der Topic Maps begründet. Topics, die als Typen anderer Objekte auftreten, sind in der Regel nicht mit anderen Topics durch Associations verbunden.	nicht behebbar
PF.CN.06	Beim Drucken wird keine Darstellung der Associations (mit Angabe der Rollen) mit ausgedruckt.	Die Druckfunktion in Scholion WB+ wurde allgemein verbessert.	behoben
PF.CN.07	Für den Druckvorgang öffnet sich ein zweites Fenster, in dem der Druckbefehl ein zweites Mal gegeben werden muss.		
UF.CN.01	Die Darstellung von Topics und zugeordneten Occurrences (z.B. Dokumente) überlappen sich gegenseitig. (Siehe LF.CN.04)	Dieser Fehler trat vorübergehend auf. Der Frame, in dem Occurrences angezeigt werden, wurde wieder an die richtige Stelle platziert.	behoben
UF.CN.02	Im Nicht-Vollbild-Modus des Internet Explorers muss der Benutzer sehr viel scrollen.	Die Aufteilung des für die GUI zur Verfügung stehenden Raumes wurde verbessert.	behoben
UF.CN.03	Beim Klicken auf „Schlagwort-Suche" erscheint keine Schnittstelle für die Benutzung der Suche.	Die Schlagwort-Suche steht nun direkt über ein Texteingabefeld unter dem Wort „Schlagwort-Suche" zur Verfügung.	behoben
UF.CN.04	Das Zurück-Navigieren aus der Suchfunktion arbeitet nicht erwartungskonform.	Siehe UF.ALLG.01 (Abschnitt 8.5 Allgemeine Erkenntnisse, Tabelle 14 auf Seite 312).	offen
UF.CN.05	Der angezeigte Navigationspfad (History besuchter Topics) berücksichtigt ein „Zurück Gehen" nicht.	Beim Zurück-Navigieren wird der letzte Eintrag aus der History besuchter Topics entfernt.	behoben
UF.CN.06	Eine Richtung von Associations ist aus	Das Topic Map Paradigma sieht ausdrück-	nicht

[37] Der Begriff „große Topic Map" bezeichnet eine Topic Map, in der eine Zahl von mindestens mehreren Hundert Topics enthalten bzw. gespeichert sind.

	der Darstellung an der GUI nicht erkennbar.	lich ungerichtete Kanten vor (vgl. dazu [Widhalm, 2002], [Pepper, 2002], [XTM, 2001]).	behebbar
UF.CN.07	Eine Funktionalität für einfache Suche wird noch nicht zur Verfügung gestellt.	Als Möglichkeit zur einfachen Suche wurde die Schlagwort-Suche zur Verfügung gestellt (vgl. UF.CN.03).	behoben

8.3 Testergebnisse: Topic-Editor

Auf den kommenden Seiten werden die Fehler beschrieben, die durch den Black-Box Test der Contentpool-Verwaltung im **Werkzeug „Topic-Editor"** (vgl. Abbildung 52: Architektur – Scholion WB+ Contentpool-Verwaltung, Seite 153, zentral im oberen Teil der Abb.) aufgezeigt werden konnten. Die folgende Tabelle 12 gibt außerdem Auskunft darüber, ob, und wenn ja, wie ein Fehler bereits behoben werden konnte.

Die hier vorgestellten Ergebnisse des Black-Box Tests wurden aus [Wizany, 2003] (S. 3 – 28) entnommen. Bei einzelnen Fehlern kann es sein, dass Maßnahmen zur Behebung und Status des Fehlers nicht mit dem Quelldokument überein stimmen. In solchen Fällen wurde nach Fertigstellung des Dokuments von B. Wizany noch eine Fehlerkorrektur durchgeführt.

Tabelle 12: Fehler im Topic-Editor

Klassifikation	Kurze Beschreibung	Maßnahme(n) zur Behebung	Status
LF.TE.01	Beim Erstellen eines neuen Topics werden unnötige Scroll-Balken angezeigt.	Die Scroll-Balken scheinen durch einen Fehler in der XSL-Transformation zu Stande zu kommen. Die genaue Ursache ist aber noch unbekannt.	offen
LF.TE.02	Im Formular für das Erstellen eines neuen Topics werden unterschiedliche Schriftarten bzw. -größen verwendet.	Die Felder des Formulars wurden korrigiert.	behoben
PF.TE.01	Der Button „Zurück zu Schritt 2" beim Erstellen eines Topics führt zum Abbrechen des Vorgangs.	Hinter den Button wurde nun die richtige Aktion gelegt.	behoben
PF.TE.02	Das Löschen automatisch generierter Topics ist nicht möglich.	Die automatisch generierten Topics werden durch das TM4J Framework erzeugt (vgl. [TM4J, 2002]).	nicht behebbar
PF.TE.03	Tritt beim Löschen von Topics ein Fehler auf, so werden keine Details der Fehlermeldung angezeigt.	Der Benutzer erhält nun detailliertes Feedback über die aufgetretene Fehlersituation.	behoben
PF.TE.04	Alphanumerische Werte bei der Eingabe einer textbasierten Occurrence werden als Verknüpfung (URI) interpretiert.	Die automatische Erkennung von URI's wurde verbessert.	behoben
UF.TE.01	Der Vorgang zum Speichern neuer Topics dauert sehr lange.	Der Fehler hing mit einem Fehler bei der Datenübertragung, nicht mit den Algorithmen im Topic-Editor zusammen.	behoben

UF.TE.02	Bei Wahl des Menüpunkts „Wissensbasis navigieren“[38] verschwinden die Menüpunkte des Topic-Editors.	Der Link wurde umbenannt in „Zum Content-Navigator wechseln“. Dadurch ist der Wechsel zu einem anderen Werkzeug für den Benutzer offensichtlich.	behoben
UF.TE.03	Tritt beim Importieren einer Topic Map ein Fehler auf, so fehlt das Feedback an den Benutzer.	Die Bearbeitungslogik für den Import einer Topic Map wurde überarbeitet. Trotzdem konnte der Fehler nicht gefunden werden.	offen
UF.TE.04	URI’s in Occurrences, bei denen die Angabe des Protokolls (z.B. „http://“) fehlt, werden als relative Adressangaben zur Adresse der Contentpool-Verwaltung interpretiert.	Die automatische Erkennung von URI’s wurde verbessert.	behoben

8.4 Testergebnisse: Topicmap-Manager

Auf den kommenden Seiten werden die Fehler beschrieben, die durch den Black-Box Test der Contentpool-Verwaltung im **Werkzeug „Topicmap-Manager“** (vgl. Abbildung 52: Architektur – Scholion WB+ Contentpool-Verwaltung, Seite 153, im rechten oberen Teil der Abb.) aufgezeigt werden konnten. Die folgende Tabelle 13 gibt außerdem Auskunft darüber, ob, und wenn ja, wie ein Fehler bereits behoben werden konnte.

Die hier vorgestellten Ergebnisse des Black-Box Tests wurden aus [Wizany, 2003] (S. 3 – 28) entnommen. Bei einzelnen Fehlern kann es sein, dass Maßnahmen zur Behebung und Status des Fehlers nicht mit dem Quelldokument überein stimmen. In solchen Fällen wurde nach Fertigstellung des Dokuments von B. Wizany noch eine Fehlerkorrektur durchgeführt.

Tabelle 13: Fehler im Topicmap-Manager

Klassifikation	Kurze Beschreibung	Maßnahme(n) zur Behebung	Status
LF.TM.01	Die Namen der Menü-Einträge sind teilweise zu lang.	Im Menü wird nun eine kleinere Schriftart verwendet. Zudem wurde der Rahmen, in dem das Menü dargestellt ist, verbreitert.	behoben
UF.TM.01	Bei (angeblich) fehlerhaften Importen von Topic Maps erschien die Topic Map anschließend doch im System.	Rückmeldung über Erfolg bzw. Misserfolg nach dem Import einer Topic Map stimmen nun mit dem tatsächlichen Resultat des Vorgangs überein.	behoben
UF.TM.02	Die Funktion „Umbenennen einer Topic Map“ ist mit einer in die Irre führenden Bezeichnung belegt.	Die Bezeichnung der Funktion wurde auf „Topic Map umbenennen“ geändert.	behoben
UF.TM.03	Beim Anlegen einer neuen Topic Map wird automatisch ein leeres Topic ge-	Das leere Topic wird automatisch durch das TM4J Framework erzeugt (vgl. [TM4J,	nicht behebbar

[38] Der Befehl „Wissensbasis navigieren” wechselt in das Werkzeug „Content-Navigator”.

	neriert.	2002]).	
UF.TM.04	Alle standardmäßig generierten Topics[39] sollten am Ende aller Übersichts- und Auswahl-Listen angezeigt werden, um die Darstellung übersichtlicher werden zu lassen.	Alle Topics werden nach wie vor nach dem Alphabet sortiert angezeigt. Es ist nicht klar, welcher Ansatz zu bevorzugen ist.	offen

8.5 Allgemeine Erkenntnisse

Auf den kommenden Seiten werden die allgemeinen Fehler beschrieben, die durch den Black-Box Test der Contentpool-Verwaltung aufgezeigt werden konnten. Mit „allgemeinen Fehlern" sind Fehler gemeint, die in der Contentpool-Verwaltung auftreten, aber **nicht eindeutig** einem der drei Werkzeuge (vgl. Abbildung 52: Architektur – Scholion WB+ Contentpool-Verwaltung, Seite 153) logisch **zugeordnet** werden können. Vielmehr sind dies Fehler, die die Contentpool-Verwaltung als Ganzes betreffen. Die folgende Tabelle 14 gibt außerdem Auskunft darüber, ob, und wenn ja, wie ein Fehler bereits behoben werden konnte.

Die hier vorgestellten Ergebnisse des Black-Box Tests wurden aus [Wizany, 2003] (S. 3 – 28) entnommen. Bei einzelnen Fehlern kann es sein, dass Maßnahmen zur Behebung und Status des Fehlers nicht mit dem Quelldokument überein stimmen. In solchen Fällen wurde nach Fertigstellung des Dokuments von B. Wizany noch eine Fehlerkorrektur durchgeführt.

Tabelle 14: Allgemeine Fehler in den Werkzeugen der Contentpool-Verwaltung

Klassifikation	Kurze Beschreibung	Maßnahme(n) zur Behebung	Status
UF.ALLG.01	Der Back-Button des Internet Explorers funktioniert nicht erwartungskonform.	Der Fehler liegt mit der Architektur von Scholion WB+ zusammen.[40]	nicht behebbar
UF.ALLG.02	Ein momentaner Status der Contentpool-Verwaltung (z.B. aktives Tool) ist an der GUI nicht ersichtlich.	Der an der GUI noch zur Verfügung stehende Raum ist begrenzt. Es konnte noch keine Lösung gefunden werden.	offen
UF.ALLG.03	Standard-Buttons in Formularen können nicht mit der „Enter"-Taste ausgelöst werden.	Dieser Fehler kommt durch die XSL-Transformation zu Stande. Es wurde noch keine Ausbesserung durchgeführt.	offen
UF.ALLG.04	Bei Operationen, die eine lange Berechnungszeit benötigen, wird unzureichendes Feedback an den Benutzer gegeben.	Aufgrund der Architektur von Scholion WB+ lässt sich schwer erkennen oder voraussagen, wann eine Operation längere Zeit dauert oder dauern wird. Ein Ansatz zur Lösung ist daher schwierig zu	offen

[39] Standardmäßig generierte Topics sind jene, die im XTM Standard als so genannte core concepts definiert wurden (vgl. [XTM, 2001]).

[40] Manche Inhalte in Scholion WB+ werden periodisch neu geladen. Auf diese Weise wird z.B. die Funktion des Online Messenging ermöglicht. Für Details dazu vgl. [Fürlinger, 2003].

		finden.	
UF.ALLG.05	Die in Windows üblichen Funktionen zum Kopieren und Einfügen von Text funktionieren in den Formularen der Contentpool-Verwaltung nicht.	Der Fehler liegt mit der Architektur von Scholion WB+ zusammen (vgl. die Ausführungen zu UF.ALLG.01).	nicht behebbar

8.6 Zusammenfassung

Im Kapitel 8 wurde durch funktionale Tests sichergestellt, dass die implementierten Werkzeuge die im Design definierten Anforderungen erfüllen. Die meisten der gefundenen Fehler wurden korrigiert.

Durch das Testen der implementierten Software wurden folgende Ziele der Diplomarbeit erfüllt:

- Die gewünschten Eigenschaften des Datenmodells (Interoperabilität und Offenheit bei minimalem Speicherplatzbedarf und minimalen Zugriffszeiten) wurden in den Tests festgestellt.
- Die korrekte Funktionsweise der Werkzeuge für die Erzeugung von Wissensatomen und für den Zugriff auf die Wissensatome im Contentpool wurde geprüft.
- Dadurch wurde erreicht, dass Lehrende beim Prozess des Lehrens und Lernende beim Prozess des Lernens besser unterstützt werden als ohne Vorhandensein eines Contentpools.

9 Fazit und Ausblick

Ziel der vorliegenden Diplomarbeit war die **Erweiterung der bestehenden Funktionalität** von Wissenstransfer-Umgebungen durch die Entwicklung des Konzepts für einen Contentpool. Der Contentpool sollte dazu dienen, Wissenstransfer-Umgebungen mit einem Pool aus zerlegten elektronischen Lehrmaterialien auszustatten. Um zu zeigen, wie die Inhalte (d.h. die Wissensatome) des Contentpools genutzt werden können, sollten weiters die Architekturen für Werkzeuge zum Zugriff auf die Inhalte des Contentpools definiert werden.

Zu diesem Zweck wurden zuerst Anforderungen an den Contentpool auf Grundlage des aktuellen Standes der Erkenntnis (v.a. in den Fachgebieten „Content Management" und „semantische Netze") ermittelt. Anschließend wurde eine Architektur für den Contentpool definiert, Anforderungen für Werkzeuge zum Zugriff auf den Contentpool wurden ermittelt und Werkzeug-Architekturen designt. Schließlich wurden die entwickelten Werkzeug-Architekturen mit einer Implementierung in Java beispielhaft in die Realität umgesetzt.

Als **Fazit** lässt sich feststellen, dass nicht alle, aber viele der gesteckten Ziele dieser Diplomarbeit erreicht werden konnten. Welche der Ziele erreicht wurden und welche nicht, ist detailliert im Kapitel 9.1 weiter unten beschrieben. Die Gründe für das **Nichterreichen von Zielen** sind vor allem:

- Die zeitlichen und personellen Ressourcen im Projekt Scholion WB+ waren begrenzt.
- Der für die Realisierung gewisser Ziele erforderliche Zeitaufwand wurde zu Beginn des Projekts zu niedrig eingeschätzt.

Die **globale Fragestellung** der Diplomarbeit lautete wie folgt: Wie müssen Lehrmaterialien zerlegt werden, um unabhängig von ihrem ursprünglichen Kontext in einem neuen Zusammenhang dargestellt oder zitiert werden zu können? Wie kann die Speicherung zerlegter Lehrmaterialien optimal erfolgen, so dass dabei die semantischen Zusammenhänge nicht verloren gehen?

Folgende der detaillierten Fragestellungen konnten **nicht beantwortet** werden:

- Das automatische Extrahieren von semantischen Zusammenhängen bei der Zerlegung von Dokumenten wurde nicht gelöst.
- Die werkzeuggestützte Erstellung neuer Dokumente aus Wissensatomen wurde nicht realisiert.

Folgende der detaillierten Fragestellungen wurden **beantwortet**:

- Die Zerlegung von elektronischen Dokumenten wurde gelöst.
- Für die Speicherung von zerlegten Dokumenten und semantischen Zusammenhängen wurde ein Datenmodell spezifiziert.
- Die Darstellung von Wissensatomen an einer Benutzungsschnittstelle wurde realisiert.
- Eine Funktion zur Suche in den Wissensatomen wurde implementiert.

Im Rest des Kapitels wird exakt berichtet, welche der in den Kapiteln 1 (Einleitung) und 2 (Ausgangssituation und Problemstellung) gesteckten Ziele wie erreicht werden konnten und welche Ziele warum nicht erreicht werden konnten (Abschnitt 9.1: Bilanzierung der Zielerreichung). An Hand dieser „**Ziel-Bilanz**" wird schließlich angeschnitten, welche Forschungs- und Implementierungsarbeiten offen bleiben bzw. durch die vorliegende Diplomarbeit aufgeworfen werden (Abschnitt 9.2: Future Work).

9.1 Bilanzierung der Zielerreichung

In den ersten beiden Kapiteln der Diplomarbeit wurden die Ziele und Fragestellungen der Diplomarbeit **vom Groben zum Feinen** („top down") definiert und gruppiert. Genau so wird auch im vorliegenden Abschnitt die Zielerreichung zuerst im Groben (auf Ebene von übergeordneten Zielen und Fragestellungen) diskutiert und danach im Detail auf einzelne Ziele und Fragen eingegangen.

Die Ziele für die Diplomarbeit wurden in Abschnitt 1.1 (Seite 2 ff.) definiert. Sie sind als Rahmenziele für die Beantwortung von Fragestellungen der Diplomarbeit zu sehen. Die Ziele der Diplomarbeit wurden wie folgt erreicht:

- Ob die **Effizienz des Wissenstransfers** und die **Wiederverwendbarkeit der Lernmaterialien** erhöht werden konnten, kann nur empirisch beantwortet werden. Empirische Daten, die Aussagen zu diesen Zielen liefern könnten, sind aber noch nicht vorhanden.
- Lehrende (Professoren und Lehrbeauftragte) sowie Lernende (Studierende) haben mit der Realisierung des Contentpools mit Sicherheit mehr Funktionalität zur Verfügung als bisher. Ob dadurch die Lehrenden und Lernenden bei der **Erfüllung ihrer jeweiligen Aufgaben** besser unterstützt werden, kann gleichfalls nur empirisch ermittelt werden.
- Die restlichen **Rahmenziele** konnten erfüllt werden, wie später gezeigt wird.

Im Abschnitt 2.3 (Ziele für das Design der Software, Seite 18 ff.) des Kapitels 2 wurden Ziele für die Werkzeuge der Contentpool-Verwaltung wie folgt gruppiert: Allgemeine Benutzbarkeitskriterien, Nutzung von Content, Speicherung von Content und Aufbereitung von Content. Die Ziele für Design und Implementierung der Software wurden wie folgt erreicht:

- Allgemeine **Benutzbarkeitskriterien** wurden bei der Implementierung der Werkzeuge beachtet. Bisher fehlen aber empirische Daten, die bestätigen oder widerlegen könnten, dass die Benutzbarkeitskriterien erfüllt wurden.
- Zur **Nutzung von Content** wurden Werkzeuge für die Verwaltung des Contentpools implementiert. Das Betrachten von gespeicherten Wissensatomen an der Benutzungsschnittstelle der Wissenstransfer-Umgebung wird unterstützt. Funktionalität zum Erstellen von Kursmaterialien und Nachschlagewerken konnte dagegen erst konzeptuell entworfen werden.
- Für die **Speicherung von Content** werden die Werkzeuge der Contentpool-Verwaltung ebenfalls genutzt. Die Werkzeuge bieten viel Funktionalität für das Erstellen, Bearbeiten und Löschen von semantischen Netzen und Wissensatomen.

Die Speicherung von Wissensatomen in verschiedenen Codalitäten ist bis jetzt erst in Ansätzen realisiert.

- Die **Aufbereitung von Content** wird mit dem Splitter-Algorithmus, der in einem Werkzeug implementiert wurde, unterstützt. Der Dokument-Splitter stellt ebenfalls (wie die anderen Werkzeuge der Contentpool-Verwaltung) die vorgesehene Funktionalität in einer ersten, ausbaufähigen Version zur Verfügung. Alle Werkzeuge bieten noch genügend Spielraum für Verbesserungen und Erweiterungen.

In den folgenden Abschnitten werden die obigen einleitenden Ausführungen bis auf Ebene einzelner Ziele konkretisiert.

9.1.1 Allgemeine Ziele der Diplomarbeit

Viele der in Abschnitt 1.1 (Seite 2 ff.) definierten Ziele beziehen sich auf Messgrößen, die ohne eine empirische Benutzer-Befragung nicht zu ermitteln sind. Eine **Evaluierung der Zufriedenheit** von Benutzern der Wissenstransfer-Umgebung Scholion WB+ wurde von E. Mielach im Zeitraum Oktober 2002 bis März 2003 durchgeführt [Mielach, 2003]. Da der Fokus der Evaluierung auf der Wissenstransfer-Umgebung als Ganzes lag, sind für die Beurteilung, ob die allgemeinen Ziele der Diplomarbeit realisiert werden konnten, nur sehr begrenzt empirische Daten vorhanden.

- Die **Effizienz des Wissenstransfers** konnte durch den Einsatz von Scholion WB+ in der Lehre erhöht werden [Mielach, 2003]. Ein kausaler Zusammenhang mit der Verwendung des Contentpools besteht jedoch nicht. Damit bleibt unklar, ob die Effizienz des Wissenstransfers mit Hilfe des Contentpools erhöht wird.
- Aufgrund fehlender empirischer Befunde bleibt auch unklar, ob die **Wiederverwendbarkeit von Lernmaterialien** erhöht werden konnte. Sicher ist: einmal erstellte Lernmaterialien können für beliebig viele Kurse in Scholion WB+ eingesetzt werden. Dies gilt unabhängig davon, ob der Contentpool für die Erstellung der Lernmaterialien eingesetzt wird. Zur Beurteilung, ob das Ziel der höheren Wiederverwendbarkeit erreicht wurde, möge folgende **These** dienen: „Durch die Verwendung des Contentpools kann die Wiederverwendbarkeit mit hoher Wahrscheinlichkeit weiter erhöht werden, da die Wahrscheinlichkeit dafür sinkt, dass neue Wissensatome während der Erstellung von Kursmaterialien erstellt werden müssen.“[41]
- Das Ziel, **Lernende** beim Prozess des Lernens besser zu unterstützen, wurde nicht erreicht. Es wurden nur Voraussetzungen geschaffen, um dieses Ziel zu erreichen. Lernende werden in Scholion WB+ beim Prozess des Lernens durch die Möglichkeit, persönliche **Annotationen und Hervorhebungen** zu erstellen, un-

[41] „…die Wahrscheinlichkeit (…) sinkt, dass neue Wissensatome (…) erstellt werden müssen.“ Mit dieser Aussage ist die Annahme verbunden, dass die Zahl der Wissensatome im Contentpool mit der Zeit ansteigt. Aus einer höheren Anzahl von Wissensatomen ergibt sich eine geringere Wahrscheinlichkeit, dass Wissensatome zu gesuchten Themen nicht gefunden werden und folglich erstellt werden müssten. Je geringer die Zahl von während der Erstellung eines Kursmaterials neu erstellten Wissensatome ist, desto höher muss die Wiederverwendbarkeit sein.

terstützt [Fürlinger, 2003]. In den Werkzeugen zum Zugriff auf den Contentpool fehlt solche Funktionalität. Durch die Implementierung des Contentpools werden Lernende folglich noch **nicht** bei ihren Aufgaben **unterstützt**.

- Das Ziel, **Lehrende** beim Prozess des Lehrens besser zu unterstützen, wurde nicht erreicht. Es wurden nur Voraussetzungen geschaffen, um dieses Ziel zu erreichen. Für Lehrende wird in Scholion WB+ ein **Editor zum Erstellen von Kursmaterialien** zur Verfügung gestellt [Fürlinger, 2003]. Dieses Editor-Werkzeug bietet zur Zeit noch keine Möglichkeit, auf den Contentpool zuzugreifen, es arbeitet also isoliert. Bis jetzt wurden im Contentpool lediglich Schnittstellen zum Zugriff definiert. Durch die Implementierung des Contentpools werden Lehrende folglich noch **nicht** bei ihren Aufgaben **unterstützt**.
- Der *state of the art* für **relevante wissenschaftliche Fachgebiete** wurde als Grundlage für die entwickelten Algorithmen, Konzepte und Architekturen verwendet. Zusätzlich wurde Erkenntnis aus kommerziellen Produkten der E-Learning-Märkte berücksichtigt.
- Die Ziele für die **Spezifikation des Datenmodells** wurden erfüllt: Interoperabilität und Offenheit wurden durch Berücksichtigung wichtiger Standards erreicht. Durch das Design des Contentpools wurden Speicherbedarf und Zugriffszeit minimal gehalten.
- Ein Werkzeug für die **Erzeugung von Wissensatomen** wurde implementiert. Bei der Funktionalität des Werkzeugs musste aber auf die automatische Generierung von semantischen Zusammenhängen verzichtet werden.
- Werkzeuge für den **Zugriff auf den Contentpool** wurden implementiert. Ein Werkzeug zur Unterstützung der Generierung neuer Dokumente wurde aber nicht realisiert.

9.1.2 Allgemeine Benutzbarkeitskriterien

Durch die Anwendung anerkannter **ergonomischer Richtlinien** wurde versucht, den allgemeinen Benutzbarkeitskriterien so weit wie möglich Rechnung zu tragen. Die **Zufriedenheit der Benutzer** mit der entstandenen Software konnte aus Mangel an Zeit und personellen Ressourcen noch nicht gemessen werden. Durchgeführt wurde bereits eine Evaluierung der Zufriedenheit mit Scholion WB+ allgemein [Mielach, 2003]. Die genannte Untersuchung geht jedoch nicht auf die Nutzung des Contentpools ein.

Aus diesem Grund liegen keine Befunde vor, die eine Entscheidung zulassen, ob das Ziel der Beachtung allgemeiner Benutzbarkeitskriterien erfüllt wurde.

9.1.3 Ziele für die Nutzung von Content

Die Ziele für die Nutzung von Content (Inhalten) konnten weitgehend durch die Funktionalität der implementierten Werkzeuge erfüllt werden:

- Es wurde eine mächtige Funktionalität zur **Suche in den Wissensatomen** des Contentpools implementiert. Bei der Suche können semantische Zusammenhänge berücksichtigt werden, wobei der Suchvorgang bei einzelnen Schlagwörtern be-

ginnt. Durch die Umsetzung von für die Anforderungen des Contentpools adaptierten **Hypertext-Konzepten** wird dem Benutzer ermöglicht, semantische Verknüpfungen als „Navigationspfade" zu nutzen. Mit anderen Worten: eine Navigation nach dem Gesichtspunkt semantischer Zusammenhänge wird durch den Content-Navigator unterstützt.

- Ein Werkzeug zum **Erstellen von Kursmaterialien und Nachschlagewerken** wurde erst in Ansätzen implementiert. Bis jetzt besitzt das Werkzeug jedoch noch prototypischen Charakter. Die bisherige Funktionalität lässt reichlich Spielraum für Verbesserungen und zusätzliche Features. In der jetzigen Version müssen Lernmaterialien ausschließlich manuell editiert werden (vgl. dazu [Fürlinger, 2003]).
- Durch die Einbindung von **Multimedia-Inhalten** wird der Forderung nach **beliebiger Codalität** der Inhalte Rechnung getragen. Dieses Ziel konnte mit Einschränkungen erreicht werden. Bis jetzt ist nur die Einbindung von Grafiken oder Text in Wissensatome möglich.
- Das **Exportieren von Inhalten** aus dem Contentpool ist in Ansätzen ebenfalls realisiert. Der Export ist bis jetzt noch auf das XTM Format (vgl. [XTM, 2001]) beschränkt.
- **Semantische Beziehungen** zwischen Wissensatomen können mit Hilfe des Topic-Editors erstellt, bearbeitet und gelöscht werden. Dem Benutzer wird hierbei ein großes Spektrum an Funktionalität angeboten.
- Das Erstellen, Bearbeiten und Löschen von Wissensatomen zum Zweck der Pflege des im Contentpool abgebildeten Wissens ist ebenfalls unter Zuhilfenahme des Topic-Editors möglich.
- Die **Wiederherstellung eines Originaldokuments** aus den Wissenatomen wird durch das verwendete Datenmodell ermöglicht. Ein Wissensatom wird stets zusammen mit Information zu seiner Position im Originaldokument gespeichert.

9.1.4 Ziele für die Speicherung von Content

Für die Speicherung von Content wurde ein offenes und erweiterbares Datenmodell spezifiziert. Beide genannten Eigenschaften konnten durch die bedeutende Rolle, die den XML-Technologien (wie z.B. den XML Topic Maps) zugemessen wurde, in hohem Ausmaß sichergestellt werden. Das Datenmodell weist folgende Eigenschaften auf.

- Inhalte (Wissensatome) und das semantische Netz mit Wissen über die Inhalte werden **getrennt gespeichert**. Teilweise ist die Grenze fließend; d.h. manche Teile der Inhalte werden innerhalb eines semantischen Netzes, aber in eigenen Objekten gespeichert. Die Rede ist von Occurrences mit textbasiertem Inhalt (vgl. dazu Abschnitt 4.4.2.3.4 Verbindung bzw. Vorkommen („Occurrences"), Seite 89).
- **Metadaten** zu den Wissensatomen können auf Grundlage des entworfenen Datenmodells sehr **umfangreich** abgespeichert werden (z.B. Autor, Titel des Dokuments). Dadurch wird auch eine **Versionsnummerierung** ermöglicht.

- Eine Schnittstelle für die effiziente Suche im Contentpool wurde implementiert. Diese **Such-Schnittstelle** wird wie weiter oben beschrieben genutzt.

9.1.5 Ziele für die Aufbereitung von Content

Die Aufbereitung von Inhalten zu Wissensatomen wird im Dokument-Splitter realisiert.

- Das **automatische Zerlegen** von Dokumenten konnte für die Formate Text-Datei, HTML-Datei und PDF-Datei verwirklicht werden. Durch die Konfiguration von gewünschten Größen der erzeugten Wissensatome werden gute Ergebnisse der Zerlegung erzielt.
- Die automatische **Erkennung von semantischen Zusammenhängen** konnte nicht realisiert werden. Semantische Zusammenhänge werden ausschließlich durch manuelles Editieren erfasst. Dieser Schritt ist nicht synchron mit der Zerlegung möglich.
- Eine **automatische Aufbereitung** der Wissensatome mit Metadaten konnte erst in Ansätzen verwirklicht werden.
- Für die **manuelle Nachbearbeitung** der Zerlegung und der Metadaten bietet der Dokument-Splitter ein mächtiges Instrumentarium an. Alle geforderten Möglichkeiten konnten realisiert werden: Verschmelzen zu kleiner, Zerteilen zu großer Wissensatome; Verschieben der Zerteilungsgrenzen (d.h. der Unterteilung eines Dokuments in Wissensatome); manuelle Bearbeitung der Metadaten.
- Das **automatische Auffinden von semantischen Beziehungen** konnte bis jetzt noch nicht realisiert werden. Neben dem Mangel an zeitlichen Ressourcen scheiterte die Erreichung dieses Ziels vor allem an der großen Komplexität dieser Aufgabe.

9.2 Future Work

Aus der Bilanzierung der Ziele dieser Arbeit ergibt sich der Bedarf für **weitere Forschungen** oder für die Weiterentwicklung der implementierten Werkzeuge der Contentpool-Verwaltung. Einerseits sind empirische Befunde über die Auswirkungen des neuen Contentpool-Konzepts zu erbringen. Andererseits sind die Möglichkeiten der bereits implementierten Werkzeuge noch nicht ausgeschöpft. Daraus ergibt sich weiterer Entwicklungsbedarf. Beide Themen werden in der Folge behandelt.

9.2.1 Fehlende empirische Befunde

Für manche Ziele der Diplomarbeit kann nicht beantwortet werden, ob sie erreicht wurden. Die Ursache dafür ist in fehlenden empirischen Befunden zu sehen. Folgende Befunde sind zu erbringen:

- Durch eine Benutzer-Befragung sowohl von Lehrenden als auch von Lernenden ist die Frage zu klären, ob die **Effizienz des Wissenstransfers** durch den Einsatz eines Contentpools erhöht werden kann.

- Empirische Untersuchungen müssen zeigen, ob die **Wiederverwendbarkeit von Lernmaterialien** durch den Einsatz eines Contentpools verbessert (d.h. erhöht) werden kann. Diese Untersuchungen müssen überwiegend mit den Lehrenden durchgeführt werden.
- Die Beachtung von allgemeinen Benutzbarkeitskriterien muss durch **Usability-Tests** geprüft werden. Die Usability-Tests müssen sowohl mit Lehrenden als auch Lernenden der Wissenstransfer-Umgebung durchgeführt werden.

9.2.2 Weiterer Implementierungsbedarf

Ein Teil der Fragestellungen der Diplomarbeit konnte **theoretisch beantwortet** werden. Zur praktischen Umsetzung, d.h. zur Implementierung in einem Werkzeug, fehlten die zeitlichen und personellen Ressourcen. Die noch nicht in einem Werkzeug realisierten Funktionen sind folgende:

- Es ist ein Werkzeug zu implementieren, das den **Lernenden** erlaubt, ihre Lernprozesse in der Wissenstransfer-Umgebung unter Einbeziehung des Contentpools effizienter durchzuführen. Das Werkzeug muss die **Erstellung von Nachschlagewerken** durch Lernende aus Wissensatomen des Contentpools unterstützen.
- Die **Integration** des Contentpools mit dem **Editor-Werkzeug** (das ist das Werkzeug für die Erstellung von Kursmaterialien und Nachschlagewerken – vgl. [Fürlinger, 2003]) ist zu verbessern. Bis jetzt ist die Einbindung von Inhalten aus dem Contentpool für den Benutzer mit einem hohen Zeitaufwand verbunden. Damit wird das Ziel verfolgt, **Lehrende** bei ihren Lehrprozessen in der Wissenstransfer-Umgebung besser zu unterstützen.
- Die Funktionalität der Werkzeuge der Contentpool-Verwaltung ist so zu unterstützen, dass die **Forderung nach Multi-Codalität** der Inhalte besser erfüllt werden kann. Konkret ist es wünschenswert, dass auch Audio- oder Videoinhalte im Contentpool gespeichert und in elektronischen Unterlagen eingebunden werden können.
- Für den **Export von Inhalten** aus dem Contentpool ist ein Werkzeug zu entwickeln, das in geeigneter Weise die Speicherung der Inhalte in einem Dokument im PDF-Format ermöglicht. In diesem Zusammenhang ist die Frage von Bedeutung, wie die **Darstellung und Anordnung** der Inhalte geschehen kann, so dass der Nutzen der erzeugten Dokumente möglichst hoch sein wird. Je besser die Lernenden bei der Erreichung ihrer Lernziele von den erzeugten Dokumenten unterstützt werden, desto höher ist deren Nutzen.
- Noch ist die **Integration des Dokument-Splitters** mit den übrigen Werkzeugen der Contentpool-Verwaltung (Content-Navigator, Topic-Editor, Topicmap-Manager) gering. Durch eine verstärkte Integration kann die Benutzbarkeit aller Werkzeuge noch wesentlich verbessert werden.

Manche Fragestellungen der Diplomarbeit konnten gar nicht beantwortet werden. Um sie zu klären, sind Lösungen auch auf konzeptueller Ebene zu entwickeln. Eine Implementie-

rung in Werkzeugen kommt im Anschluss an die Entwicklung der konzeptuellen Lösung in Betracht. Folgende Fragen blieben unbefriedigend gelöst:

- Es sind Algorithmen zu entwickeln, mit deren Hilfe die **semantischen Beziehungen** zwischen Wissensatomen, die mit Dokument-Splitter erzeugt wurden, effizient und sinnvoll automatisch gefunden werden können. Werden die Algorithmen in Werkzeugen implementiert, muss eine Integration mit den bereits im Contentpool gespeicherten semantischen Netzen möglich sein.
- Die **automatische Aufbereitung** mit Metadaten von Wissensatomen, die mit Dokument-Splitter erzeugt werden, ist noch auszubauen. Bislang werden erst die wichtigsten Metadaten extrahiert. Es sind insbesondere Algorithmen für die effiziente Beschlagwortung von Wissensatomen noch zu entwickeln.
- Die Unterstützung der **Zerlegung von PDF-Dokumenten** hat sich als weit problematischer, als im Vorhinein angenommen, heraus gestellt. So konnte das Extrahieren von Grafiken oder anderen multi-codalen Elementen aus PDF-Dateien noch nicht realisiert werden. Die Verbesserung der Algorithmen, die die Inhalte aus PDF-Dateien lesen und extrahieren, ist anzustreben.

10 Anhang

In diesem Kapitel ist zusätzliche, für diese Diplomarbeit relevante Information angeführt. Zu Beginn des Kapitels wird die verwendete Literatur (Abschnitt 10.1 Referenzen) ausgewiesen. Innerhalb des Fließtexts der Diplomarbeit werden die am linken Blattrand angegebenen Kurzbezeichnungen verwendet.

Weiters sind für das Verständnis der Inhalte dieser Diplomarbeit wichtige Termini definiert (Abschnitt 10.2 Glossar).

10.1 Referenzen

[AICC, 2001] AICC CMI **Guidelines for Interoperability** (CMI001). Version 3.5, 2. April 2001.
Includes AICC Course Structure Format, AICC CMI Data Model.

[AICC, 2004] Aviation Industry CBT Comitee – Frequently Asked Questions (AICC FAQ). Online available at:
http://www.aicc.org/pages/aicc_faq.htm
[Zuletzt aufgerufen am 28. Apr 2004]

[Adobe, 2002] Adobe Systems, Inc.: **Adobe Acrobat Viewer for Java.**[42] Online available at:
http://www.adobe.com/products/acrviewer/main.html
http://itext.sourceforge.net/downloads/pdfviewer.zip
http://www.adobe.com/products/acrviewer/eula.html
[Links jeweils zuletzt aufgerufen am 28. Nov 2002]

[Auinger, 1999] Auinger Andreas: **Scholion Redevelopment (Part I)**.
Diploma Thesis, Institute of Business Informatics, University of Linz, 1999.

[Auinger, 2002] Auinger Andreas: **„Scholion WB+"** - die Zukunft der virtuellen Ausbildung. Projektplanung und Projektauftrag für die Implementierung der Wissenstransfer-Umgebung Scholion WB+.
Universität Linz, Juni 2002.

[AuingStary, 2003] Auinger Andreas, Stary Christian: **Verknüpfung von Content und Kommunikation für selbstgesteuerten, webbasierten Wissenstransfer**. Mensch & Computer 2003, GI und ACM german Chapter, Tagungsband Teubner Verlag, 2003, S.359 – 369.

[Berger, 2003] Berger Stefan: **Benutzerhandbuch Contentpool Verwaltung**. Benutzerdokumentation von Content-Navigator 0.4.3, Topic-Editor 0.5.1, Topicmap-Manager 0.4.0.
Projektstudium, Universität Linz, 2003.

[42] Adobe Systems, Inc., discontinued development and distribution of Pdf Viewer as of March, 2003. All documents mentioned above within the citation can be received from the author on request.

[Blumstengel, 1998] Blumstengel Astrid: **Entwicklung hypermedialer Lernsysteme**. Dissertation, Paderborn, 1998.
http://dsor.uni-paderborn.de/de/forschung/publikationen/blumstengel-diss/Multimedia.html
[zuletzt aufgerufen am 14. Jul 2002]

[Bortz, 1993] Bortz, Jürgen: **Statistik für Sozialwissenschaftler.**
4., vollständig überarbeitete Auflage, Berlin, 1993.

[Collier, 2002] Collier Geoff: **E-Learning Application Infrastructure**. Sun Microsystems White Paper, January 2002.
http://www.sun.com/edu/whitepapers/pdf/eLearning_Application_Infrastructure_wp.pdf
[zuletzt aufgerufen am 31. Jul 2002]

[Comp, 2002] F-Com Wissensmanagementsysteme GmbH: **Systemkomponenten- und Architekturbeschreibung der COMpendis® Software**.
http://www.sbg.ac.at/filcom/system.htm
[zuletzt aufgerufen am 08. Jul 2002]
http://www.f-com.at/f-com.wissen-managen/Seite%201.htm
[zuletzt aufgerufen am 11. Jul 2002]

[Dahn, 2000a] Dahn B. Ingo: **Automatic Textbook Construction and Web Delivery** in the 21st Century.
Appeared in: the Journal of Structural Learning Theory, 2000.

[Dahn, 2000b] Dahn B. Ingo: **Symbiose von Buch und Internet**.
In: Tagungsband LearnTec 2000, pp. 551 – 558. Wien, 2000.

[Dahn, 2000c] Dahn B. Ingo: **Logic Programming for Courseware Generation** in Slicing Book Technology.
Unofficial working paper, CADE 2000, Koblenz, 2000.

[Dahn, 2001a] Dahn B. Ingo: **Slicing Book Technology** – Providing Online Support for Textbooks.
Proceedings of the ICDE (International Conference on Distance Education), Düsseldorf, April 2001.

[Dahn, 2001b] Dahn B. Ingo, Schwabe Gerhard: **Personalizing Textbooks with Slicing Technologies** – Concept, Tools, Architecture, Collaborative Use.
Proceedings of the HICCS (Hawaii International Conference on System Sciences), Hawaii, 2001.

[Dahn, 2001c] Dahn B. Ingo: **Using Networks for Advanced Personalization of Documents**.
Proceedings of SSGRR (Scuola Superiore G. Reiss Romoli), International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet; L'Aquila, Italy, 2001.

http://www.ssgrr.it/en/ssgrr2001/papers/Ingo%20Dahn.pdf
[zuletzt aufgerufen am 07. Mai 2003]

[Dahn, 2001d] Dahn B. Ingo: **Data and Metadata in the Trial-Solution Project**, Version 1.1.
Unofficial working paper, Koblenz, 2001.

[Dahn et al., 2001] Dahn B. Ingo, Armbruster Michael, Furbach Ulrich, Schwabe Gerhard: **Slicing Books – The Authors' Perspective**.
Unpublished Manuscript prepared for: Bromme R., Stahl E.: Writing Hypertext and Learning. Conceptual and Empirical Approaches.
Koblenz, 2001.

[Dodds, 2001a] Dodds Philip V.W. (Ed.): Sharable Content Object Reference Model (SCORM) Version 1.2 – **The SCORM Overview**.
Advanced Distributed Learning Initiative (ADL) White Paper,
1 October, 2001.
http://www.adlnet.org/ADLDOCS/Document/SCORM_1.2_Overview.pdf
[zuletzt aufgerufen am 05. Aug 2002]

[Dodds, 2001b] Dodds Philip V.W. (Ed.): Sharable Content Object Reference Model (SCORM) Version 1.2 – **The SCORM Content Aggregation Model**.
Advanced Distributed Learning Initiative (ADL) White Paper,
1st October, 2001.
http://www.adlnet.org/ADLDOCS/Document/SCORM_1.2_CAM.pdf
[zuletzt aufgerufen am 05. Aug 2002]

[Dodds, 2001c] Dodds Philip V.W. (Ed.): Sharable Content Object Reference Model (SCORM) Version 1.2 – **The SCORM Run-Time Environment**. Advanced Distributed Learning Initiative (ADL) White Paper, 1 October, 2001.
http://www.adlnet.org/ADLDOCS/Document/SCORM_1.2_RunTimeEnv.pdf
[zuletzt aufgerufen am 05. Aug 2002]

[DoddsWest, 2002] Dodds Philip V.W., West Jerry: **Demystifying SCORM** – Impact and Value of the Latest E-Learning Specifications and Standards (IEEE, IMS and AICC).
Advanced Distributed Learning Initiative (ADL) Presentation,
April 10th, 2002.
http://www.adlnet.org/ADLDOCS/Other/DemystifyingSCORM.zip
[zuletzt aufgerufen am 07. Aug 2002]

[Downes, 2000] Downes Stephen: **Learning Objects**.
Essay, University of Alberta, Canada, 2000.
http://www.atl.ualberta.ca/downes/naweb/Learning_Objects.doc
[zuletzt aufgerufen am 09. Mai 2003]

[Ellmer et. al., 1998] Michael Ellmer, Thomas Pilsl: **SCHOLION – interactive telelearning system**.
Diploma Thesis, Institute of Business Informatics, University of Linz, 1998.

[ExpExch, 2003] http://www.experts-exchange.com/Programming/Programming_Languages/Java/Q20627261.html
[zuletzt aufgerufen am 10. Sep 2003]

[Fenneberg, 2002] Fenneberg Peter: **Design des idealen e-Learning Systems**. Diplomarbeit am Institut für Wirtschaftsinformatik, Schwerpunkt Communications Engineering, Universität Linz, 2003.

[Fletcher, 2001] Fletcher J.D.: **Evidence for Learning from Technology-Assisted Instruction**.
In: O'Neil H.F. jr., Perez R. (Eds.): Technology Applications in Education – A Learning View. Lawrence Erlbaum Associates, Hillsdale, NJ, 2001.

[Froschauer et al., 1999] Froschauer Barbara, Stary Christian, Bramböck Franz, Ellmer Michael, Ortner Wolfgang, Totter Alexandra: **SCHOLION** - Scaleable Technologies for Telelearning.
University of Linz, 1999.

[Fürlinger, 2003] Fürlinger Stefan: **Annotationen und webbasiertes Lernen**. Diplomarbeit am Institut für Wirtschaftsinformatik, Schwerpunkt Communications Engineering, Universität Linz, 2003.

[Garshol, 2002] Garshol, Lars Marius: **The Linear Topic Map Notation**. Definition and Introduction, Version 1.2. Ontopia AS Whitepaper, Oslo (Norway), May 2002. Online available at:
http://www.ontopia.net/download/ltm.html
[zuletzt aufgerufen am 29. Aug. 2002]

[Grüln, 2002] Grülnberger Wolfgang: **Modellierung eines COMpendis-Beispiels in einer assoziativen Datenbank**.
Projektstudium, Institut für Wirtschaftsinformatik, Universität Linz, 2002.

[HeinRoith, 1998] HEINRICH Lutz J., Roithmayr Friedrich: **Wirtschaftsinformatik-Lexikon**. 6. Auflage, München, 1998.

[Hibernate, 2003] Bradby Daniel, Andersen Max R., King Gavin, Bauer Christian, et al.: **Hibernate – Object/Relational Mapping and Transparent Object Persistence for Java**. Homepage of the Hibernate Project.
http://hibernate.org/
http://sourceforge.net/projects/hibernate
[Links jeweils zuletzt aufgerufen am 03. Nov 2003]

[IMS Content, 2001] IMS Global Learning Consortium: **IMS Content Packaging Information Model**. Version 1.1.2, Final Specification, 2001.

http://www.imsproject.org/content/packaging/cpv1p1p2/imscp_infov1p1p2.html
[zuletzt aufgerufen am 10. Jul 2002]

[IMS Cont XML, 2001] IMS Global Learning Consortium: **IMS Content Packaging XML Binding**. Version 1.1.2, Final Specification, 2001.
http://www.imsproject.org/content/packaging/cpv1p1p2/imscp_bindv1p1p2.html
[zuletzt aufgerufen am 10. Jul 2002]

[IMS CP, 2002] IMS Global Learning Consortium: **IMS Content Packaging Specification, XML Schema Definition**, Version 1.1.3, July 2002.
Online available at:
http://www.imsglobal.org/content/packaging/cpv1p1p3/validation/xml_schema/imscp_v1p1p3.xsd
[zuletzt aufgerufen am 28. Aug 2002]

[IMS Ent, 2002] IMS Global Learning Consortium: **IMS Enterprise Information Model**. Version 1.1, Final Specification, 01 July, 2002.
http://www.imsglobal.org/enterprise/entv1p1/imsent_infov1p1.html
[zuletzt aufgerufen am 02. Aug 2002]

[IMS Ent XML, 2002] IMS Global Learning Consortium: **IMS Enterprise XML Binding**. Version 1.1, Final Specification, 01 July, 2002.
http://www.imsproject.org/enterprise/entv1p1/imsent_bindv1p1.html
[zuletzt aufgerufen am 02. Aug 2002]

[IMS General, 2002] IMS Global Learning Consortium: **General Information**. Frequently Asked Questions (FAQ).
http://www.imsproject.org/faqs/imsfaqs.cfm
[zuletzt aufgerufen am 30. Jul 2002]

[IMS Learner, 2001] IMS Global Learning Consortium: **IMS Learner Information Packaging Information Model**. Version 1.0, Final Specification, 2001.
http://www.imsproject.org/profiles/lipinfo01.html
[zuletzt aufgerufen am 02. Aug 2002]

[IMS MD, 2002] IMS Global Learning Consortium: **IMS Learning Resource Metadata Specification, XML Schema Definition**, Version 1.1.2, November 2001.
Online available at:
http://www.imsglobal.org/xsd/imsmd_v1p2p2.xsd
[zuletzt aufgerufen am 28. Aug 2002]

[IMS Meta, 2001] IMS Global Learning Consortium: **IMS Learning Resource Meta-Data Information Model**. Version 1.2.1, Final Specification, 2001.
http://www.imsproject.org/metadata/imsmdv1p2p1/imsmd_infov1p2

p1.html
[zuletzt aufgerufen am 10. Jul 2002]

[IMS Meta XML, 2001] IMS Global Learning Consortium: **IMS Learning Resource Meta-Data XML Binding**. Version 1.2.1, Final Specification, 2001.
http://www.imsproject.org/metadata/imsmdv1p2p1/imsmd_bindv1p2p1.html
[zuletzt aufgerufen am 10. Jul 2002]

[IMS QTI, 2002] IMS Global Learning Consortium: **IMS Question and Test Interoperability ASI Information Model**. Version 1.2, Final Specification, 11 February, 2002.
http://www.imsproject.org/question/qtiv1p2/imsqti_asi_infov1p2.html
[zuletzt aufgerufen am 02. Aug 2002]

[ISO 10744/1997] International Organization for Standardization (ISO), International Electrotechnical Commission (IEC): ISO/IEC 10744 – Information Technology – **Hypermedia / Time-based Structuring Language (HyTime).** 2nd Edition.
ISO, Genf, 1997.
http://www.y12.doe.gov/sgml/wg8/document/n1920/pdf/n1920.pdf
[zuletzt aufgerufen am 08. May 2003]

[ISO 13250/1999] International Organization for Standardization (ISO), International Electrotechnical Commission (IEC): ISO/IEC 13250 Information Technology – SGML Applications – **Topic Maps**. Information Technology Document Description and Processing Languages. First Draft, Genf, 1999.
http://www.y12.doe.gov/sgml/sc34/document/0129.pdf
[zuletzt aufgerufen am 09. Jul 2002]

[ISO 13250/2000] International Organization for Standardization (ISO), International Electrotechnical Commission (IEC): ISO/IEC 13250 Information Technology – SGML Applications – **Topic Maps** – Information Technology Document Description and Processing Languages. Second Edition, Genf, 19 May 2002.[43]
http://www.y12.doe.gov/sgml/sc34/document/0322_files/iso13250-2nd-ed-v2.pdf
[zuletzt aufgerufen am 29. Aug 2002]

[iText, 2003] Lowagie Bruno, Soares Paolo: **iText – a Java PDF Library**. Online available at:
http://www.lowagie.com/iText/
[zuletzt aufgerufen am 15. Nov 2003]

[43] Note: as one can see, the second edition of this ISO/IEC Standard was published within the year 2002. Nonetheless, the official name of this document remained "ISO/IEC 13250:2000". That is the reason why I refer to it this way.

[Jadele, 2000] Ginzinger Alice, Sametinger Johannes, et al.: **Jadele (Java Based Algorithm Description Language)** – Jana (Java Based Abstract Notation for Algorithms).
University of Linz, Institute of Business Informatics, Department of Software Engineering, 2000.

[Java, 2002] Java Homepage von Sun Microsystems, Inc.:
http://java.sun.com [02. Oct 2002]

[JavaTut, 2003] The Java Tutorial: **How to use Threads**. Online available at:
http://java.sun.com/docs/books/tutorial/uiswing/misc/threads.html
[zuletzt aufgerufen am 14. Nov 2003]

[JPedal, 2003] IDRsolutions: **JPedal – Java Pdf Extraction Decode Access Library**. Pdf text and image extraction and conversion software. East Peckham, Kent (United Kingdom), 2002. Online available at:
http://www.jpedal.org/ [zuletzt aufgerufen am 16. Nov 2003]

[JUnit, 2002] Gamma Erich, Meade Erik, Beck Kent et al.: **JUnit – Testing Resources for Java Extreme Programming**. A simple framework for writing and running automated tests. Version 3.8.1, September 2002. Online available at:
http://www.junit.org/index.htm
http://sourceforge.net/projects/junit/
[Links jeweils zuletzt aufgerufen am 18. Nov 2003]

[Karrer, 2000] Karrer Wolfgang: **Scholion Redevelopment (Part II)**.
Diploma Thesis, Institute of Business Informatics, University of Linz, 2000.

[Kessler, 1993] Kessler Gary C.: **What does ITU TSS stand for?** Notice on the Discussion Board of the University of Indiana, Bloomington, IN.
http://cell-relay.indiana.edu/mhonarc/cell-relay/1993-Apr/msg00086.html
[zuletzt aufgerufen am 10. Juni 2003]

[Koper, 2000] Koper Rob: **From Change to Renewal** – Educational Technology Foundations of Electronic Environments.
Open University of the Netherlands, Educational Technology Expertise Center, 2000; latest version available at:
http://eml.ou.nl/introduction/articles.htm
[zuletzt aufgerufen am 07. Mai 2003]

[Koper, 2001] Koper Rob: **Modeling Units of Study from a Pedagogical Perspective** – the Pedagogical Meta-model behind EML.
First draft, version 2, June 2001.
Available at: http://eml.ou.nl/introduction/articles.htm
[zuletzt aufgerufen am 08. Jul 2002]

[Lange, 2001] Lange Carola: **Program Slicing and Slicing Book Technology** – a Comparison.

ARTI 8800, University of Georgia and University Koblenz-Landau, 2001.

[LOM, 2000] IEEE Draft Standard for **Learning Object Metadata (LOM) Specification**.
IEEE Learning Technology Standards Committee, Ref. no. P1484.12/D6.1. New York, 2001.

[Mielach, 2003] Mielach Edmund: **Evaluierung didaktischer Konzepte im Online-Learning**. Eine empirische Feldstudie zu Scholion WB+. Diplomarbeit am Institut für Wirtschaftsinformatik, Schwerpunkt Communications Engineering, Universität Linz, 2003.

[Noy et al., 2001] Noy Natalya F., Sintek Michael, Decker Stefan, Cruzéby Monica, Fergerson Ray W., and Musen Mark A.: **Creating Semantic Web Contents with Protégé-2000**.
In: IEEE Intelligent Systems Vol. 16(2), pp 60 – 71, Stanford, 2001.
Online available at: http://www-smi.stanford.edu/pubs/SMI_Reports/SMI-2001-0872.pdf
[zuletzt geöffnet am 28. Aug 2002]

[Oestereich, 1997] Oestereich Bernd: **Objektorientierte Software-Entwicklung** - Softwareentwicklung mit der Unified Modeling Language. 3., aktualisierte Auflage (Oldenbourg), München, Wien 1997.

[Ontopia Gen, 2002] Ontopia AS: **General Information** and Frequently Asked Questions. Ontopia AS, Oslo, Norway.
http://www.ontopia.net/index.html
[zuletzt aufgerufen am 28. Aug 2002]
http://www.ontopia.net/topicmaps/faq.html
[zuletzt aufgerufen am 28. Aug 2002]

[Park, 2002] Park Jack (Editor) and Hunting Sam (Technical Editor): **XML Topic Maps** – creating and using Topic Maps for the Web. Pearson Education Inc., Boston (MA), 2002.

[Paulsen, 2003] Paulsen, Morten Flate: **E-Learning – The State of The Art**. Work Package One Of The European Delphi Project. NKI Distance Education, March 2003. Online available at:
http://home.nettskolen.nki.no/~morten/artikler/State_of_the_art.pdf
[zuletzt aufgerufen am 16. Dez. 2003]

[PdfBox, 2002] Litchfield, Ben: **PdfBox – a Java PDF Library**. A project which will allow access to all of the components in a PDF document. Online available at:
http://www.pdfbox.org/
http://www.csh.rit.edu/~ben/projects/pdfbox/index.html
[Links jeweils zuletzt aufgerufen am 15. Nov 2003]

[PdfSpec, 2001] Adobe Systems, Inc.: **PDF Reference**. Adobe Portable Document Format, Version 1.4 Specification, third edition. Online available at:

http://partners.adobe.com/asn/developer/acrosdk/docs/filefmtspecs/PDFReference.pdf [zuletzt aufgerufen am 6. März 2003]
http://www.tex.uniyar.ac.ru/doc/PDFReference.pdf
[zuletzt aufgerufen am 16. Nov 2003]
http://www.foo.be/docs-free/PDF-ref/PDFReference.pdf
[zuletzt aufgerufen am 16. Nov 2003]

[Pepper, 2002] Pepper Steve: **The TAO of Topic Maps** – Finding the Way in the Age of Infoglut.
Ontopia AS, Oslo, 2002.
http://www.ontopia.net/topicmaps/materials/tao.html
[zuletzt aufgerufen am 12. Jul 2002]

[Radmayr, 2003] Radmayr Markus: **Web-basierte Kommunikation und Kollaboration zur Unterstützung konstruktivistischen Lernens**. Diplomarbeit am Institut für Wirtschaftsinformatik, Schwerpunkt Communications Engineering, Universität Linz, 2003.

[RechPom, 1999] Rechenberg Peter, Pomberger Gustav (Hrsg.) et al.: **Informatik-Handbuch**. Zweite, aktualisierte und erweiterte Auflage; München, Wien, 1999.

[Reindl, 2001] Reindl Klaus: **Evaluation of CAT/CAL Environments**.
Diploma Thesis, Institute of Business Informatics, University of Linz, 2001.

[RFC2396, 1998] Berners-Lee T., Fielding R., Irvine U.C., Masinter L. et al.: **Uniform Resource Identifiers (URI) – Generic Syntax**.
Network Working Group, MIT/LCS, August 1998.
http://www.ietf.org/rfc/rfc2396.txt
[zuletzt geöffnet am 14. Apr 2003]

[RobCol, 2002] Robson Robby, Collier Geoff: **E-Learning Interoperability Standards**.
Sun Microsystems White Paper, January 2002.
http://www.sun.com/edu/whitepapers/pdf/eLearning_Interoperability_Standards_wp.pdf
[zuletzt aufgerufen am 31. Jul 2002]

[Robson, 2001] Robson Robby: **E-Learning Technology Standards** – Status and Direction.
Eduworks Corporation Conference Presentation, 16 December 2001.
http://www.eduworks.com/web/Docs/Status_and_Direction.ppt
[zuletzt aufgerufen am 05. Aug 2002]

[RTD, 2002] **European Commission – Research**: Information on the European Community for Research and Technological Development (RTD).
Brussels, 2002.
http://europa.eu.int/comm/research/rtdinfo_en.html
[zuletzt aufgerufen am 11. Jul 2002]

http://europa.eu.int/comm/research/rtdinfo_de.html
[zuletzt aufgerufen am 13. Jul 2002]

[Sander, 2001] Sander Peter, Lavirotte Stéphane, Broeglin Dominique: **Trial Solution System Architecture Overview**, 2001.
http://www.trial-solution.de/SysArch.htm
[zuletzt aufgerufen am 09. Jul 2002]

[SchmZuck, 2003]: Schmitz Christof, Zucker Betty: **Wissensmanagement – Schnelleres Lernen im Unternehmen**. Metropolitan Verlag, Regensburg, Berlin 2003.

[Sedge, 1991] Sedgewick, Robert: **Algorithmen**. Bonn, München, Reading (Mass.), 1991.

[Stary, 1996] Stary Christian: **Interaktive Systeme** – Software-Entwicklung und Software-Ergonomie.
2. verbesserte und erweiterte Auflage, vieweg-Verlag, Braunschweig-Wiesbaden, 1996.

[Sullivan, 2003] Sullivan Sean C: **Dynamically Creating PDFs in a Web Application**. In: OnJava, O'Reilly, 18 June 2003. Online available at:
http://www.onjava.com/pub/a/onjava/2003/06/18/dynamic_files.html
[zuletzt aufgerufen am 16. Nov 2003]

[Sun, 1999] Davidson James D., Coward Danny: **Java Servlet Specification**. Final Release, Version 2.2.
Palo Alto, Sun Microsystems, Inc. December 1999.
http://java.sun.com/products/servlet/2.2/
[zuletzt aufgerufen am 17. Apr 2003]

[Svensson, 2001] Svensson, Mats: **E-Learning Standards and Technical Specifications**. ADAPT 2001 Working Paper, Lund (Sweden), November 2001. Online available at:
http://www.centre-inffo.fr/v2/pdf/adapt/adapt2001_chap4_angl.pdf
[zuletzt aufgerufen am 20. Nov. 2003]

[TM4J, 2002] Ahmed, Kal et al.: **TM4J – Topic Maps For Java**. A project aiming for the development of robust, open-source tools for creating, manipulating and publishing topic maps. Online available at:
http://www.tm4j.org/
https://sourceforge.net/projects/tm4j/
[Links jeweils zuletzt aufgerufen am 30. Okt 2003]

[TM Strict, 2000] Pepper Steve, et al. (Ontopia AS): **HyTime Topic Map (HyTM)** based XML Type Definition (dtd), Version 1.1, 24 October 2000. Online available at: http://www.ontopia.net/download/tmstrict.dtd
[zuletzt aufgerufen am 29. Aug 2002]

[Trial, 2003] Homepage des Trial Solution Projektes – **Project Details and Results**, April 2003.

http://www.trial-solution.de/details.htm
[zuletzt aufgerufen am 05. Mar. 2004]

[TS final, 2003] Dahn B. Ingo, et al.: **Trial Solution Project – Final Report**. Contracts IST-1999-11397 and IST-2001-35447, Final Report Deliverable D15, Koblenz, 30. April 2003.
Online available at:
http://iwm.uni-koblenz.de/trialsolution/FinalReport.pdf
[zuletzt aufgerufen am 05. Mrz. 2004]

[TS Results, 2003] Homepage des Trial Solution Projektes – **Public Results from the Trial Solution Project**, April 2003.
http://iwm.uni-koblenz.de/trialsolution/
[zuletzt aufgerufen am 05. Mar. 2004]

[ValDaSch, 2001] Valerius Marianne, Dahn B. Ingo, Schwabe Gerhard: **Adaptive Bücher für das kooperative Lernen** – Anwendungen, Konzepte, Erfahrungen.
Proceedings GeNeMe (Gemeinschaft in Neuen Medien) – Workshop zu Organisation, Kommunikation und Kooperation auf der Basis innovativer Technologien, Fakultät Informatik der Technischen Universität Dresden, Dresden 2001.

[Valerius, 2001] Valerius Marianne, SIMON Anna: **Slicing Book Technology** – eine neue Technik für eine neue Lehre?
Fachberichte Informatik 08/2001, Universität Koblenz-Landau, 2001.

[W3C, 2000] Bray Tim, Paoli Jean, Sperberg-McQueen C.M., Maler Eve: **Extensible Markup Language (XML) 1.0**.
World Wide Web Consortium (W3C) Recommendation (2nd Edition), 6 October 2000.
http://www.w3.org/TR/REC-xml/
[zuletzt aufgerufen am 14. Apr 2003]

[Weidenmann, 1997a] Weidenmann B.: **Multicodierung und Multimodalität im Lernprozess**.
In: Issing L., Klimsa P. (Hrsg.): Information und Lernen mit Multimedia, 2. überarbeitete Auflage; S. 65-84; Beltz Psychologie-Verlags-Union; Weinheim, Basel; 1997.

[Widhalm, 2002] Widhalm Richard (Hrsg.), Mück Thomas: **Topic Maps – Semantische Suche im Internet.**
Xpert Press, Berlin, 2002.

[Wizany, 2003] Wizany Bernhard: **Test der Topic Map Tools von Scholion WB+**.
Teil 1 – Black Box Test. Projektstudium, Institut für Wirtschaftsinformatik, Universität Linz, 2003.
Online available at:
http://students.idv.edu/~9255269/246204_03s/Testprotokoll%20To

picMapTools.pdf
[zuletzt aufgerufen am 18. Nov 2003]

[XTM, 2001] Pepper Steve, Moore Graham: **XML Topic Maps (XTM) 1.0** – TopicMaps.org Specification.
http://www.topicmaps.org/xtm/1.0/
[zuletzt aufgerufen am 09. Jul 2002]

[XTM dtd, 2001] Pepper Steve, Moore Graham (Eds.), et al.: **XML Topic Maps (XTM) Document Type Definition (DTD)**, Version 1.2, 8 February 2001.
http://www.topicmaps.org/xtm/1.0/xtm1.dtd
[zuletzt aufgerufen am 28. Aug 2002]

10.2 Glossar

In diesem Kapitel werden wichtige Begriffe, Definitionen, Abkürzungen und Akronyme erläutert, die im Kontext der präsentierten Inhalte und Ergebnisse der Diplomarbeit bedeutend sind. Teile der Akronyme und ihrer Beschreibungen wurden aus [Collier, 2002] und [RobCol, 2002] entnommen.

Account: eine Zugangsberechtigung zum System Scholion WB+, welche sich aus Benutzernamen und Passwort zusammen setzt und den erfolgreichen Login bei Scholion WB+ erlaubt. Sowohl der Benutzername als auch das Passwort sind Zeichenketten, welche von den Administratoren von Scholion WB+ an Benutzer vergeben werden.

ADL: Akronym für Advanced Distributed Learning Initiative.[44] ADL ist ein gemeinsames Gremium des Weißen Hauses und des amerikanischen Verteidigungsministeriums (DoD – Department of Defense), dessen Aufgabe die Unterstützung, d.h. Dokumentierung, Validierung, Förderung und z.T. auch Finanzierung der Schaffung von Standards und Spezifikationen durch andere Organisationen ist.

AICC: Akronym für Aviation Industry CBT Committee. Oft wird dieses Akronym auch für die Spezifikationen verwendet, die von diesem Komitee veröffentlicht wurden.[45] AICC wurde 1988 gegründet und beschäftigt sich vorwiegend mit der Standardisierung von Produkten zur virtuellen Ausbildung von Ingenieuren und Piloten in der Luftfahrtindustrie. Teile dieser Spezifikationen sind auch für den Bereich e-Learning von Interesse. ADL und IEEE verwenden zum Teil die Veröffentlichungen des AICC als Grundlage ihrer Arbeiten.

Alphabet: Im Sinne der Theoretischen Informatik ist ein Alphabet V eine endliche Menge von Zeichen zur Darstellung von Daten. Durch das Hintereinan-

[44] Siehe auch http://www.adlnet.org.

[45] Siehe dazu http://www.aicc.org.

derschreiben von Zeichen entstehen Zeichenketten ([RechPom, 1999], Seite 90).

Ansatz: Bezeichnung für „jede systematische Vorgehensweise beim Problemlösen“ ([HeinRoith, 1998], Seite 50). Im Sinne der vorliegenden Diplomarbeit wird der Begriff „Ansatz“ als Oberbegriff für die im Kapitel 4 (Related Work, siehe Seite 35 ff.) präsentierten Standards, Spezifikationen und Projekte gebraucht. Der Begriff ist daher als Synonym zu „Konzept“ zu verstehen (vgl. Seite 339).

ANSI: Akronym für „American National Standards Institution“. ANSI ist eine in den USA tätige Organisation, die mit der Aufgabe betraut ist, Normungen auf dem Gebiet der Informations- und Kommunikationstechnologien vorzunehmen. Mitglieder des ANSI sind Repräsentanten von einschlägigen Herstellerfirmen, Laboratorien oder anderer vergleichbarer Organisationen. ANSI ist als Vertreter der USA auch Mitglied der OSI (vgl. [HeinRoith, 1998], S. 653).

ARIADNE: Akronym für Alliance of Remote Instructional Authoring and Distribution Networks for Europe.[46] ARIADNE ist eine Organisation der Europäischen Union, deren Mitglieder aus der Industrie und dem universitären Bereich stammen. Ihr Zweck ist die Veröffentlichung von Spezifikationen bzw. Tools und Diensten für e-Learning.

ASCII: Akronym für American Standard Code for Information Interchange. ASCII ist der von der CCITT genormte Code für die binäre Repräsentation von Zeichen auf dem Gebiet der Informatik. Der ASCII-Code ist ein aus 7 Bit bestehendes Alphabet, erlaubt also die Codierung von $2^7 = 128$ Zeichen. 84 der 128 möglichen Zeichen sind für die Darstellung der invarianten Schriftzeichen reserviert. Weitere 32 der 128 möglichen Zeichen dienen zur Codierung von Steuerzeichen, die sich in Übertragungssteuerzeichen, Formatsteuerzeichen, Code-Erweiterungszeichen, Gerätesteuerzeichen, Informationstrennzeichen und sonstige Steuerzeichen unterteilen lassen. Die verbleibenden 12 Zeichen symbolisieren die Sonderzeichen ([RechPom, 1999], Seite 169 ff.).

[46] Siehe dazu http://www.ariadne-eu.org

Tabelle 15: Die Zeichen des ASCII-Codes

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.asciitable.com

ASCII-Code: Siehe ASCII.

BLOB: Akronym für Binary Large Object. Die Verwendung von BLOB's ist ein Weg zur Speicherung großer Datenmengen in einer Datenbank, die mit herkömmlichen Datentypen nicht realisiert werden kann. Ein BLOB wird als Binärdaten (binary data) in einer Datenbank gespeichert. Siehe auch: CLOB. Binäre Daten verwenden den gesamten Wertebereich eines bytes zur Codierung von Daten.

Boole'scher Wert: Ein boole'scher Wert dient zur Darstellung eines der beiden logischen Werte „wahr" oder „falsch". Mit anderen Worten ist ein boole'scher Wert die Antwort auf eine Aussage, die mit „wahr" oder „falsch" beantwortet werden kann. Folglich ist ein boole'scher Wert ein Zeichen des Alphabets B = {true, false}. Mit Hilfe von logischen Operatoren können boole'sche Werte nach den Gesetzen der boole'schen Algebra verknüpft und so neue Aussagen gewonnen werden.

CAM: Akronym für Content Aggregation Model, ein Teilmodell des SCORM Referenzmodells. Siehe dazu auch SCORM.

CBT: Akronym für Computer Based Training. Man versteht darunter jede Lehrveranstaltung oder Lehreinheit, an der die Teilnahme mit Hilfe eines Einzelrechners möglich ist.

CCITT: Akronym für Comité Consultatif International Télégraphique et Téléfonique (Internationale technische Normungsbehörde der Telefongesellschaften). Heute wird nicht mehr die Bezeichnung CCITT, sondern statt dessen ITU-TSS verwendet.

CLOB: Akronym für Character Large Object. Die Verwendung von CLOB's ist ein Weg zur Speicherung großer Datenmengen in einer Datenbank, die mit herkömmlichen Datentypen nicht realisiert werden kann. Ein CLOB wird als Textdaten (character data) in einer Datenbank gespeichert. Textdaten verwenden nur einen eingeschränkten Wertebereich eines bytes (abhängig vom Code, z.B. 7 Bit bei ASCII, 2x8 Bit bei Unicode) zur Codierung der Daten.

CMI: Akronym für Computer Managed Instruction.

Codalität: Begriff, der aus der Aufspaltung des Begriffs Multimedia entsteht. Eine gute Definition findet sich bei [Blumstengel, 1998]: „[Weidenmann, 1997a], S. 65 ff. kritisiert, dass Multimedia aus mediendidaktischer Sicht zu unpräzise ist. Er schlägt vor, statt des Medienbegriffs die Kategorien Modalität und Codalität in den Vordergrund zu stellen. – Unter *Modalität* wird das wahrnehmende Sinnesorgan verstanden. Multimodale Angebote sprechen also mehrere Sinne wie Sehen, Hören, Tastsinn etc. an. Dagegen betrifft der Begriff der *Codalität* das verwendete Zeichen- bzw. Symbolsystem (verbal, piktorial, Zahlensystem u. a.). Multicodalität ist demnach die Verwendung verschiedener Formate zur Codierung von Botschaften."

Content-Management-System: ein Werkzeug bzw. eine Umgebung, mit dessen Hilfe Objekte zur Speicherung von Inhalten aller Art erzeugt, gespeichert, verwaltet, wiederverwendet und exportiert werden können.

Contentpool: Bezeichnung für die Komponente von Wissenstransfer-Umgebungen, welche in diesem Projekt erstellt werden soll und zur Speicherung zerlegter elektronischer Lehrbücher und Dokumente (= Wissensatome) dient. Der Contentpool hat im wesentlichen die Funktionalität eines Content-Management-Systems zu erfüllen. Das in der Diplomarbeit präsentierte Konzept sieht zwei Bestandteile für einen Contentpool vor: eine Datenbasis, in der die zerlegten Dokumente gespeichert sind sowie ein semantisches Netz, das zur Speicherung semantischer Beziehungen zwischen Wissensatomen dient.

Content-Verwaltung: bezeichnet den Vorgang der Erstellung, Nutzung und Wartung einer Wissensbasis. Mit diesem Begriff kann auch die Verwaltung von Wissensatomen in der Datenbasis des Contentpools gemeint sein.

Datenelement: siehe LOM Datenelement.

DBMS: Akronym für Datenbankmanagementsystem. Datenbankmanagementsysteme, oder auch Datenbankverwaltungssysteme sind Softwareprodukte, die zur dauerhaften, integren und von jedweder Anwendung un-

abhängigen Speicherung und Verwaltung von Daten dienen ([RechPom, 1999], S. 876).

DHTML: Akronym für Dynamic HTML. Dynamisches ist eine Erfindung von Marktstrategen, sagen Kritiker. In der Tat ist Dynamisches HTML keine klassische HTML-Erweiterung in Gestalt neuer HTML-Elemente. Es ist auch keine neue Sprache. Dynamisches HTML ist vielmehr der Sammelbegriff für verschiedene Lösungen, um es dem Autor einer Web-Seite zu ermöglichen, Elemente der Web-Seite während der Anzeige dynamisch zu ändern, sei es automatisch oder durch Einwirken des Anwenders.

DOM: Akronym für Document Object Model. DOM ist eine vom W3-Konsortium verabschiedete Norm, die zunächst einmal den Scriptsprachenzugriff auf beliebige Elemente eines Auszeichnungssprachen-Dokuments beschreibt. Es definiert lediglich Objekte, Eigenschaften und Methoden, die eine Scriptsprache umsetzen sollte, wenn sie sich DOM-fähig bezeichnen will.

DTD (oder dtd): Akronym für Document Type Definition, eine Beschreibungssprache für zulässige Elemente in einem SGML Dokument. Ein Dokument zur formalen Beschreibung von strukturierten Dokumenten, welches in der XML Declaration Syntax verfasst wurde, wird als DTD bezeichnet. Eine DTD regelt folglich, welche Elemente und Attribute innerhalb eines SGML Dokuments zulässig sind, in welcher Reihenfolge, und in welcher Anzahl.

ERD: Akronym für Entity Relationship Diagram, ein mit Hilfe von ERM erstelltes Diagramm zur Spezifikation eines Datenmodells.

ERM: Akronym für Entity Relationship Model. ERM (oder auch zu deutsch Gegenstands-/Beziehungsmodell) ist ein Mittel zur Erstellung eines logischen (semantischen) Modells für das Datenschema eines Datenbanksystems ([RechPom, 1999], Seite 904). Neuere Konzepte zur Modellierung von Datenschemata verwenden objektorientierte Grundsätze. Das bekannteste Beispiel hierfür ist die UML.

HTML: Akronym für Hypertext Markup Language, die am weitesten verbreitete Auszeichnungssprache. Mit der HTML werden Web-Seiten (Multimediadokumente, deren Inhalte Text, Grafik, Ton- oder Videosequenzen sein können) im World Wide Web (WWW) beschrieben ([RechPom, 1999], S. 1006).

Hypermedia: Die Integration von verschiedenen Medientypen (Grafik, Sound, Video, Text oder Sprachausgabe) in beliebiger Kombination in ein assoziatives System der Informationsspeicherung und –gewinnung ([Widhalm, 2002], S. 408).

HyTime: Abkürzung (verkürzte Bezeichnung) des ISO/IEC Standards 10744:1997 – „Hypermedia/Time-based Structuring Language". HyTime ist eine so genannte „enabling architecture" auf Basis des Datenaustauschformats SGML. Es handelt sich dabei um eine Strukturierungssprache zur Dar-

stellung von Hypermedia. Mit Hilfe von HyTime können Hypertext Linking realisiert sowie Ressourcen (Mediendateien) zeitlich und räumlich koordiniert bzw. synchronisiert werden (vgl. [Widhalm, 2002] sowie [ISO 10744/1997]).

HyTM: Akronym für HyTime Topic Map, eine offizielle Topic Map Syntax auf Basis des ISO HyTime Standards.

IEEE: Akronym für "Institute of Electrical and Electronics Engineers". IEEE ist eine weltweit tätige Gesellschaft zur Entwicklung und Förderung der Informations- und Kommunikationstechnik durch Kooperation ihrer Mitglieder. Die Mitglieder von IEEE stammen sowohl aus dem universitären Bereich als auch aus der Praxis. Von IEEE werden laufend Standards entwickelt, welche oft als ANSI-IEEE Normen zur weiteren Veröffentlichung gelangen (vgl. [HeinRoith, 1998], S. 656).

IEEE LTSC: Akronym für IEEE Learning Technology Standards Committee.[47] Das LTSC genehmigt und veröffentlicht international anerkannte Standards zur Gewährleistung von Interoperabilität zwischen e-Learning Umgebungen.

IMS: Kurzbezeichnung für IMS Global Learning Consortium, Inc.[48] Das Akronym IMS bedeutet „Instructional Management Systems". Die Verwendung der ausführlichen Schreibweise wird jedoch vom Konsortium selbst nicht mehr gewünscht, da diese oft missverstanden wurde und Irreführungen verursacht wurden (vgl. [IMS General, 2002]). Dieser Empfehlung wird im vorliegenden Dokument entsprochen und lediglich die Abkürzung verwendet.

ISO: Akronym für International Standards Organization.[49] ISO kann als das europäische Gegenstück zum IEEE angesehen werden. Aufgabe der ISO ist die Schaffung und Veröffentlichung international anerkannter, offener Standards auf Grundlage eines konsensbasierten Vorgehensmodells.

ITU-TSS: Akronym für International Telecommunications Union – Telecommunications Standards Sector (seit 1993; früher: CCITT) [Kessler, 1993].

Jadele: Akronym für Java Based Algorithm Description Language. Jadele ist eine auf Java basierende formale Beschreibungssprache für Algorithmen [Jadele, 2000].

Jana: Eine Variante der Algorithmen-Beschreibungssprache Jadele, die weitgehend von syntaktischem Ballast befreit wurde, dem Benutzer weitgehende Freiheiten erlaubt und deshalb eine besonders leicht zu erlernen-

[47] Siehe auch http://ltsc.ieee.org

[48] Siehe auch http://www.imsglobal.org

[49] Siehe auch http://www.iso.ch/iso/en/ISOOnline.frontpage

de Sprache zur Spezifikation von Algorithmen darstellt. Details der Beschreibungssprache werden in [Jadele, 2000] erläutert.

JSP: Akronym für Java Server Pages [Java, 2002].

Konsistenz: bezeichnet die Eigenschaft des Contentpools, sich in einem gültigen Zustand zu befinden. Gültig ist der Zustand genau dann, wenn es keine semantischen Widersprüche innerhalb von Wissensatomen gibt und wenn es im semantischen Netz des Contentpools keine Referenzen auf nicht mehr vorhandene Wissensatome gibt.

Konzept: Bezeichnung für den (oft mit Hilfe eines Modells) entworfenen Idealzustand eines Ausschnitts aus der Wirklichkeit. Ein Konzept ist also Vorbild für die Wirklichkeit, das in der Regel auf Grundlage eines Modells der Wirklichkeit entworfen wird ([RechPom, 1999], S. 1025 f.). „Ein Konzept beschreibt, wie die Wirklichkeit sein *soll*, es ist also immer Sollkonzept, und ein Modell beschreibt etwas, was Wirklichkeit *ist*." (L.J. Heinrich in [RechPom, 1999], S. 1025.) Der Vorgang zum Abbilden eines Ausschnitts der Wirklichkeit in ein Modell heißt „Modellieren"; der Vorgang zum Verwirklichen eines Konzepts heißt „Übertragen" (vgl. [RechPom, 1999], S. 1026).

Kursmaterial: ein aus Wissensatomen durch einen Lehrenden erzeugtes persönliches Dokument mit Assessment-Elementen. Im Kursmaterial ist die Erstellung und Speicherung von persönlichen Annotationen möglich.

Learning Object: englische Bezeichnung für ein Objekt, das sinngemäß einem Wissensatom entspricht. Die englische Originaldefinition lautet: "Learning Objects are chunks of data that are used by e-learning systems – they are authored, stored, cataloged, assembled, delivered and reported on." [Collier, 2002].

Lernunterlage: Sammelbegriff für sämtliche Materialien, die nach den Präferenzen eines Benutzers aus den im Contentpool gespeicherten Inhalten zusammen gestellt werden können, d.h. für Kursmaterialien und Nachschlagewerke.

LIP: Akronym für Learner Information Packaging, eine Spezifikation des IMS Global Learning Consortium. Hinweis: diese Spezifikation wird in der Diplomarbeit nicht behandelt.

LMS: Akronym für Learning Management System (deutsch: Lernmanagementsystem), ein Softwaresystem, innerhalb dessen Konzepte des computergestützten Wissenstransfers („Telelearning") implementiert sind; der Begriff meint also eine Wissenstransfer-Umgebung und wird daher auch synonym zu diesem Begriff verwendet.

LOM: Akronym für Learning Object Metadata. LOM ist ein Standard des IEEE LTSC für die Beschreibung von Metadaten zu Learning Objects (vgl. dazu [LOM, 2000]).

LOM Datenelement: eine Teilmenge der Learning Object Metadaten (LOM), für die der Name, eine Erklärung, die Größe, die Reihenfolge, der Wertebereich und der Datentyp eindeutig definiert sind ([LOM, 2000], S. 6).

LTSC: Siehe IEEE LTSC.

Metadaten: Allgemein „Daten über Daten", d.h. konkret im Zusammenhang mit Scholion WB+: Daten zur Beschreibung von Wissensatomen, die ein sinnvolles Auffinden der Wissensatome aus der Datenbasis des Contentpools mit Hilfe von Suchbedingungen erlauben.

MVC: Akronym für Model View Controller. MVC ist ein Konzept für die sinnvolle Behandlung von Inhalt, Layout und Verarbeitungslogik bei Webanwendungen. Die Idee hinter MVC ist die strikte Trennung von Daten (Model), Anzeige (View) und der Steuerung (Controller). Viele MVC Frameworks verwenden ein zentrales Servlet, den Front Controller, welches alle Anfragen entgegennimmt. Der Front Controller ruft die zu einer Seite gehörende Funktionalität auf und übergibt schließlich an eine Server Page, die die Aufgabe der Darstellung übernimmt. Siehe dazu Abbildung 45 (XML-spezifisches MVC-Konzept einer Web-Anwendung (Quelle: nach [Fürlinger, 2003])) auf Seite 145 sowie Abbildung 47 (Erweitertes XML-spezifisches MVC-Konzept von Scholion WB+) auf Seite 147.

Nachschlagewerk: ein aus Wissensatomen erzeugtes persönliches Dokument, bei dem keine Assessment-Elemente enthalten sind und auch keine persönlichen Annotationen erstellt werden können.

NLII: Akronym für National Learning Infrastructure Initiative.[50]

OKI: Akronym für Open Knowledge Initiative.[51] OKI ist ein Konsortium, bestehend aus akademischen Institutionen, das sowohl eigene Spezifikationen als auch Implementierungsempfehlungen veröffentlicht.

Ontologie: Bezeichnung für die ausdrückliche Spezifikation der Konzepte in einem Fachgebiet und der Beziehungen zwischen den Konzepten. Eine Ontologie stellt also ein formales Vokabular über ein Fachgebiet für den Austausch von Information zur Verfügung. ("An ontology is an explicit specification of the concepts in a domain and the relations among them, which provides a formal vocabulary for information exchange." [Noy et al., 2001].)

OSI: Akronym für "Open Systems Interconnection".

QTI: Akronym für Question & Test Interoperability, eine Spezifikation des IMS Global Learning Consortium. Hinweis: diese Spezifikation wird in der Diplomarbeit nicht behandelt.

---

[50] Siehe auch http://www.educause.edu/nlii

[51] Siehe auch http://web.mit.edu/oki/

RDBMS: Akronym für Relationales Datenbankmanagementsystem. Ein RDBMS ist ein DBMS, dessen Daten in Tabellen (Relationen) gespeichert sind.

Repository: Englischsprachiges Synonym für den Contentpool.

SBT: Akronym für Slicing Book Technology.

Scholion: Akronym für Scaleable Technologies for Teleteaching and Telelearning, eine von Dr. Andreas Auinger und Mag. Wolfgang Karrer am Institut für Wirtschaftsinformatik – Communications Engineering (vormals: „Schwerpunkt für Communications Engineering") entwickelte Wissenstransfer-Umgebung (vgl. [Auinger, 1999], [Karrer, 2000]).

SCO: Akronym für Sharable Content Object. Siehe SCORM.

SCO Referenzmodell: siehe SCORM.

SCORM: Akronym für Sharable Content Object Reference Model.[52] SCORM ist ein Referenzmodell der Regierung der Vereinigten Staaten zur Verwendung von e-Learning Standards und Spezifikationen. SCORM basiert auf den Arbeiten und Veröffentlichungen von AICC, IMS und IEEE und wurde von ADL definiert.

Semantische Einheit: eine Einheit an Information, deren semantische Bedeutung ohne zusätzliche Information erfasst werden kann. Eventuell benötigtes Vorwissen, das für das Verständnis der in einer semantischen Einheit gespeicherten Information notwendig ist, wird nicht als „zusätzliche Information" angesehen.

SGML: Akronym für Standard Generalized Markup Language, die Grundlage aller Auszeichnungssprachen (ISO 8879:1985)[53]. SGML ist der internationale Standard für die Definition von Beschreibungen für strukturierte elektronische Dokumente.

Spezifikation: Die Zusammenstellung der Anforderungen an ein Produkt (z.B. ein Softwareprodukt) und die Abbildung der Anforderungen in ein Modell bzw. das Ergebnis dieser Abbildung in Form eines Dokuments (vgl. [HeinRoith, 1998], Seite 494 f.). In der Regel ist eine Spezifikation weniger verpflichtend als ein Standard, sollte jedoch trotzdem als verbindlich angesehen werden.

SSL: Akronym für Secure Socket Layer. SSL ist eine Variante des http-Protokolls, bei dem eine zusätzliche Anwendungsschicht für die Verschlüsselung der übertragenen Daten sorgt.

Standard: Ein allgemein akzeptiertes, z.B. durch eine Norm definiertes Niveau von Anforderungen für ein genau bestimmtes Produkt (z.B. ein Softwareprodukt). Das Niveau wird als vorbildlich oder mustergültig angesehen und

[52] Siehe auch http://www.adlnet.org.

[53] Siehe http://www.iso.ch/ .

das Handeln bei der Fertigung eines Produkts danach ausgerichtet (vgl. [HeinRoith, 1998], Seite 499).

Telelearning: Bezeichnung für das Lernen unter Zuhilfenahme eines Computersystems. Charakteristisch für das Telelearning ist meist eine orts- und zeitasynchrone Lernsituation; d.h. der Prozess des Lernens ist nicht mehr an einen bestimmten Raum und/oder eine bestimmte Zeit gebunden, wie dies bei traditionellem Präsenzunterricht der Fall ist. Siehe auch „Virtueller Klassenraum".

Telelearning-Anwendung: siehe bei „Wissenstransfer-Anwendung".

Trial Solution: Akronym für Tools for Reusable, Integrated, Adaptable Learning – Systems/Standards for Open Learning Using Tested, Interoperable Objects and Networking.[54]

UML: Akronym für Unified Modeling Language, ein Quasi-Standard auf dem Gebiet von Sprachen zur Modellierung der realen Welt. UML wird in erster Linie dazu eingesetzt, ein konzeptuelles (semantisches) Modell eines Datenschemas für ein Datenbanksystem zu spezifizieren ([RechPom, 1999], S. 904).

URI: Akronym für Unified Resource Indicator. URIs sind Zeichenketten, die zur eindeutigen Identifikation von Ressourcen im Internet dienen [RFC2396, 1998]. Durch die Vergabe von URIs werden elektronische Ressourcen über eine Vielzahl von Namensräumen und Zugriffsmethoden im Netz (wie z.B. http, ftp, usw.) allgemein verfügbar gemacht.[55]

Virtueller Klassenraum: Bezeichnung für ein Lernszenario, bei dem meist eine räumliche und zeitliche Trennung zwischen der Vermittlung und der Aufnahme von Wissen erfolgt. Die Vermittlung von Wissen erfolgt durch das Bereitstellen von Lernmaterialien, auf die mit Hilfe eines Computers über ein Netzwerk zugegriffen werdden kann. Durch den Zugriff auf die Lernmaterialien wird die Aufnahme von Wissen realisiert. Die Bildung von „virtuellen Klassen" erfolgt durch die Vergabe von Zugriffsrechten auf die Lernmaterialien.

Wissensatom: ein Teil eines digitalen Dokuments, bestehend aus einem Inhaltsteil in Form einer semantischen Einheit und einem Metadaten-Teil. Wissensatome sind die kleinsten, sinnvoll zusammen hängenden Einheiten von Wissen, in die ein digitales Dokument zerlegbar ist.

Wissensbasis: Eine Sammlung von semantischen, voneinander unabhängigen Einheiten (Wissenseinheiten), deren interne Struktur durch Wissensrepräsentationsformalismen vorgegeben wird (vgl. [RechPom, 1999], Seite 978). Der Begriff ist nicht zu verwechseln mit dem Contentpool-Begriff.

---

[54] Siehe auch http://www.trial-solution.de/

[55] Vgl. auch http://www.w3.org/Addressing/ .

Wissenstransfer-Anwendung: bezeichnet Software zur Unterstützung der Übertragung von Wissen (d.h. zur Unterstützung des Lernens) in virtuellen Klassenräumen. Wie bereits aus der Bezeichnung hervorgeht, ist das Ziel von „Wissenstransfer-Anwendungen", eine bestmögliche (software-) technische Unterstützung als Voraussetzung für Transfer von Wissen eines Lehrenden zu den Lernenden oder innerhalb der Lernenden in räumlich und/oder zeitlich voneinander getrennten Situationen zu bieten.

XML: Akronym für Extensible Markup Language. XML ist eine eingeschränkte Variante der Standard Generalized Markup Language (SGML). Für die Spezifikation des XML Standards siehe [W3C, 2000].

XSL: Akronym für eXtensible Stylesheet Language.

XSLT: XSLT is designed for use as part of XSL, which is a stylesheet language for XML. In addition to XSLT, XSL includes an XML vocabulary for specifying formatting. XSL specifies the styling of an XML document by using XSLT to describe how the document is transformed into another XML document that uses the formatting vocabulary.[56]

XTM: Akronym für XML Topic Maps. (Vgl. dazu das Kapitel 4.4.2, Seite 77 ff.)

[56] Vgl. dazu http://www.w3c.org